



版权相关注意事项：

- 1、书籍版权归著者和出版社所有
- 2、本PDF来自于各个广泛的信息平台，经过整理而成
- 3、本PDF仅限用于非商业用途或者个人交流研究学习使用
- 4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
- 5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
- 6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
- 7、请于下载PDF后24小时内研究使用并删掉本PDF

版权相关注意事项：

- 1、书籍版权归著者和出版社所有
- 2、本PDF来自于各个广泛的信息平台，经过整理而成
- 3、本PDF仅限用于非商业用途或者个人交流研究学习使用
- 4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
- 5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
- 6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
- 7、请于下载PDF后24小时内研究使用并删掉本PDF

全链路学习软件开发过程
全方位掌控自动化和性能

全面梳理各阶段输入输出
全栈测试工程师入门手册

质量全面管控

从项目管理到容灾测试

葛长芝 鲁盈盈 欧仕强 编著

中国工信出版集团

电子工业出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

作者简介

葛长芝
2001年毕业于东北师范大学，有15年软件测试行业的经验，先后就职于500强企业益海嘉里、支付行业快钱、在线教育行业掌门一对一，担任测试架构师和测试总监职位，具有丰富的自动化测试和性能测试经验。

鲁盈盈
具有10年大型软件研发测试经验。曾任Infosys测试主管，有5年在美国的工作经验，关注于跨国跨区域分布式系统的性能测试、深入性能调优和JVM调优。

欧仕强
从事软件测试7年，在黑盒测试、自动化测试和专项测试方面有着丰富的经验，很早就开始参与移动端测试、接口测试和PC端测试，擅长自动化框架开发和搭建。曾就职于惠普和携程，主要负责框架开发和维护。

思维导图

质量全面管控

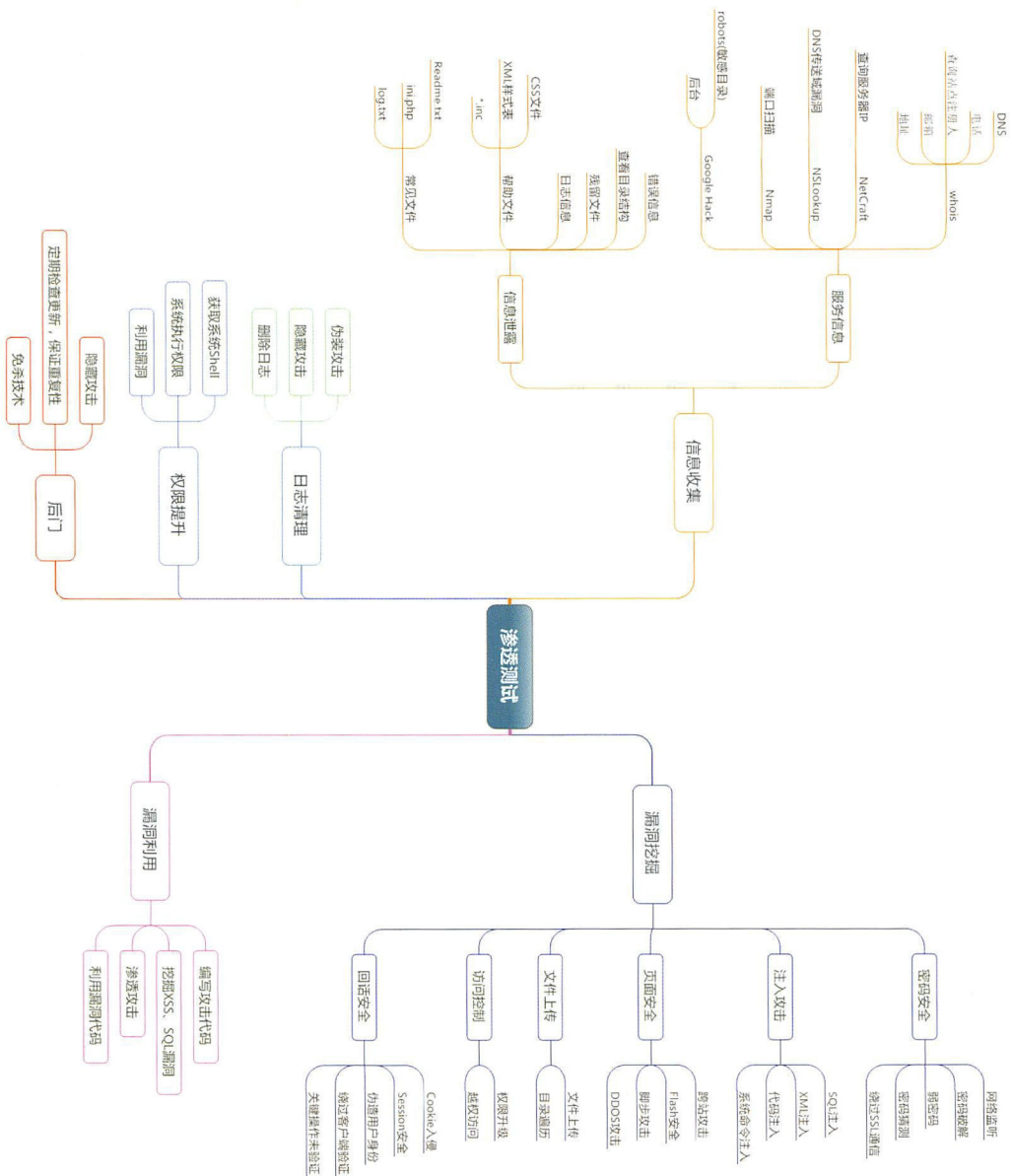
- 项目管理
 - 需求管理
 - 计划管理
 - 风险管理
 - 沟通管理
 - 质量管理
 - 人力资源管理
 - 采购管理
 - 配置管理
 - 收尾管理
- 测试管理
 - 测试策略
 - 测试计划
 - 测试用例
 - 测试执行
 - 测试报告
 - 测试工具
- 性能测试
 - 性能测试理论
 - 性能测试工具
 - 性能测试案例
- 安全测试
 - 安全测试理论
 - 安全测试工具
 - 安全测试案例

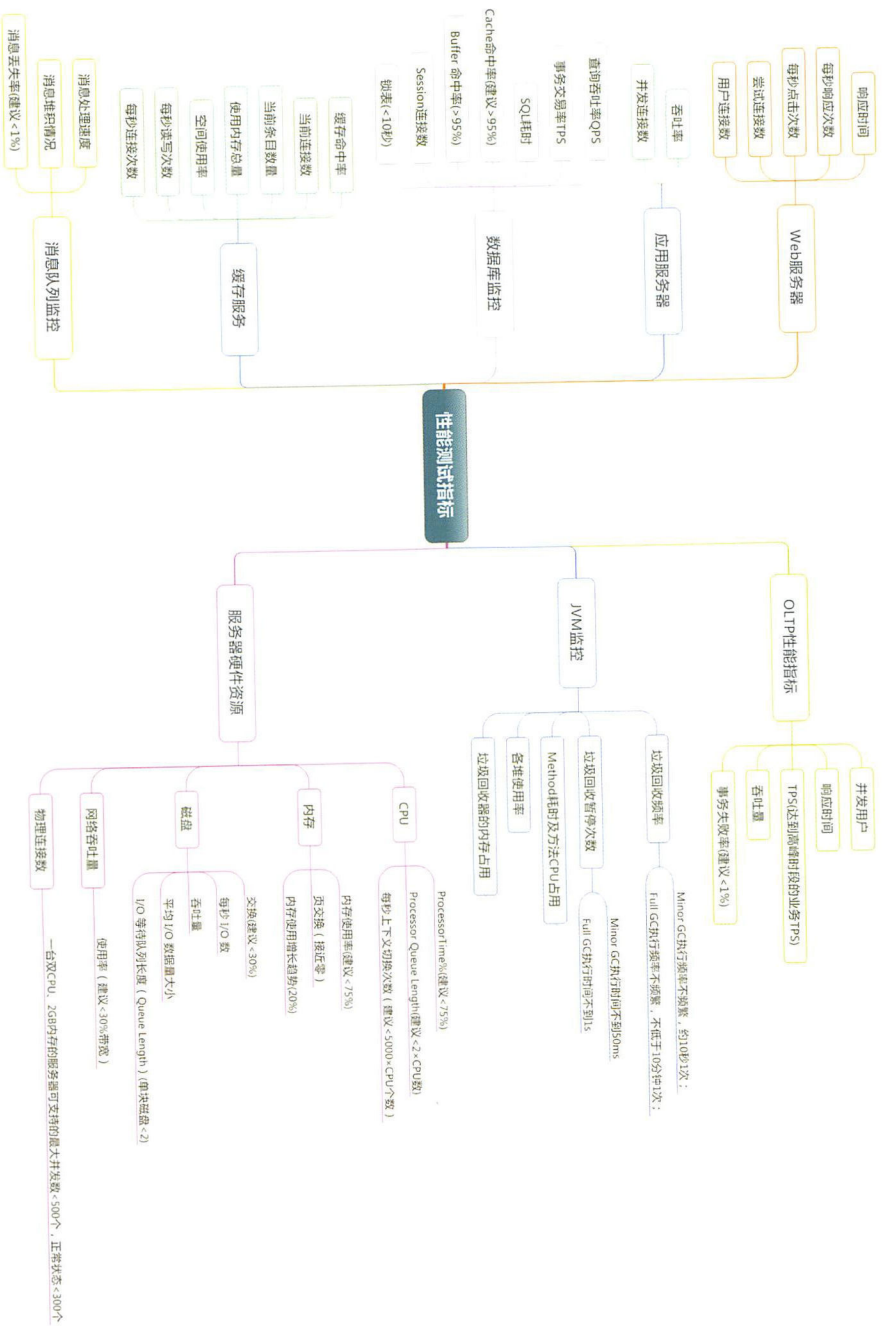
安全测试

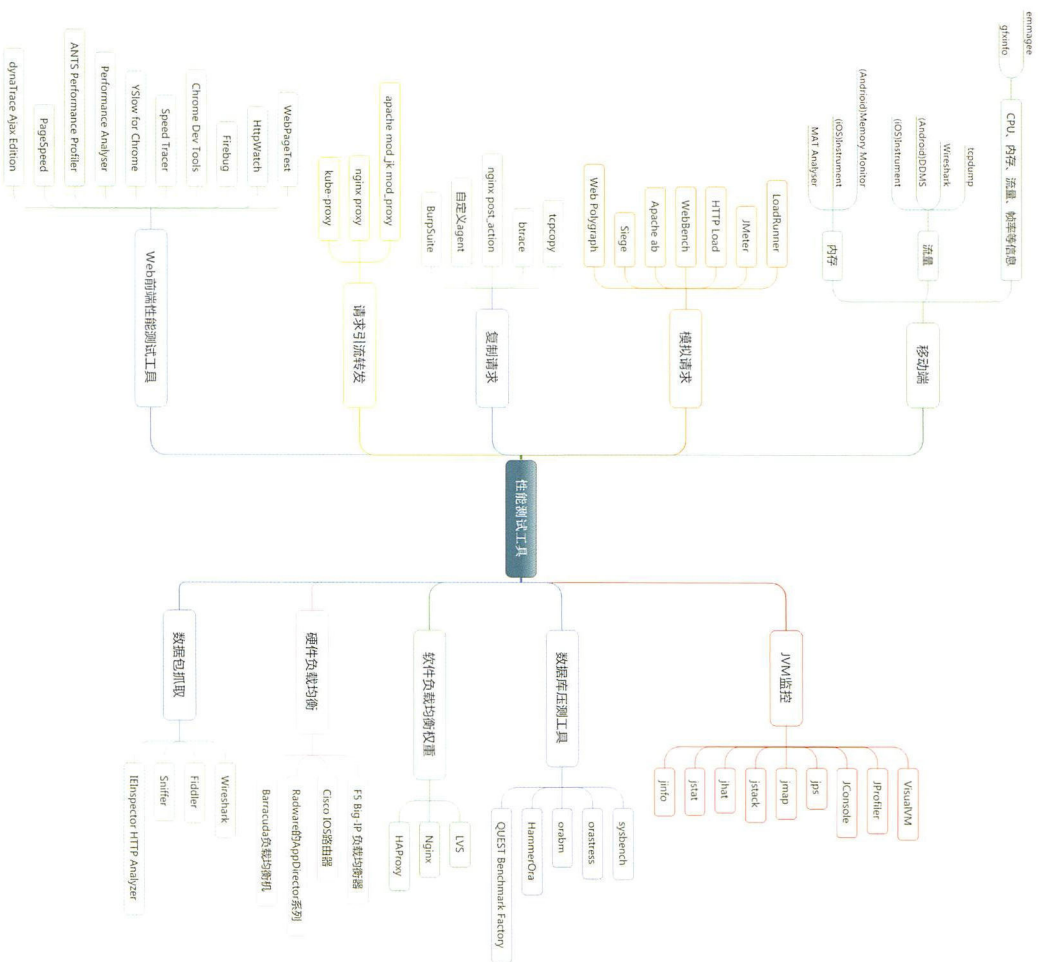
- 漏洞管理
 - 漏洞扫描
 - 漏洞分析
 - 漏洞修复
- 代码管理
 - 代码审查
 - 代码测试
 - 代码审计
- 性能测试
 - 性能测试理论
 - 性能测试工具
 - 性能测试案例
- 安全测试
 - 安全测试理论
 - 安全测试工具
 - 安全测试案例

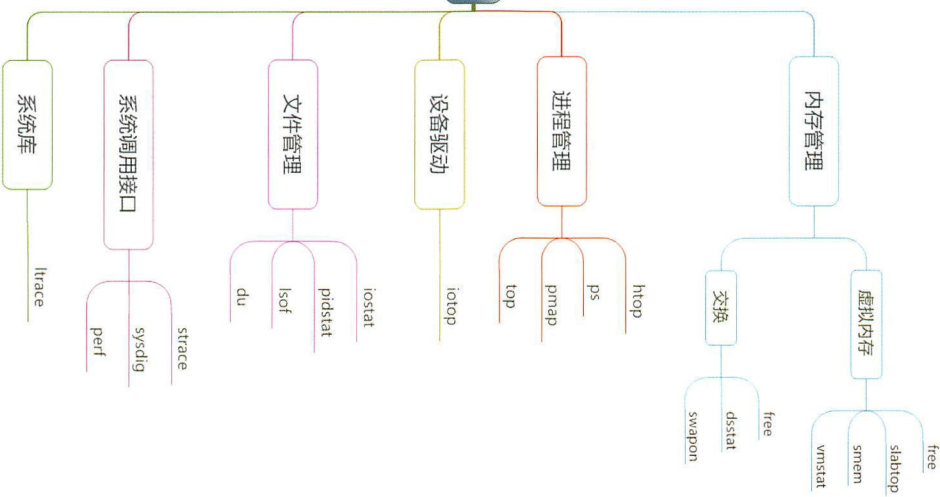
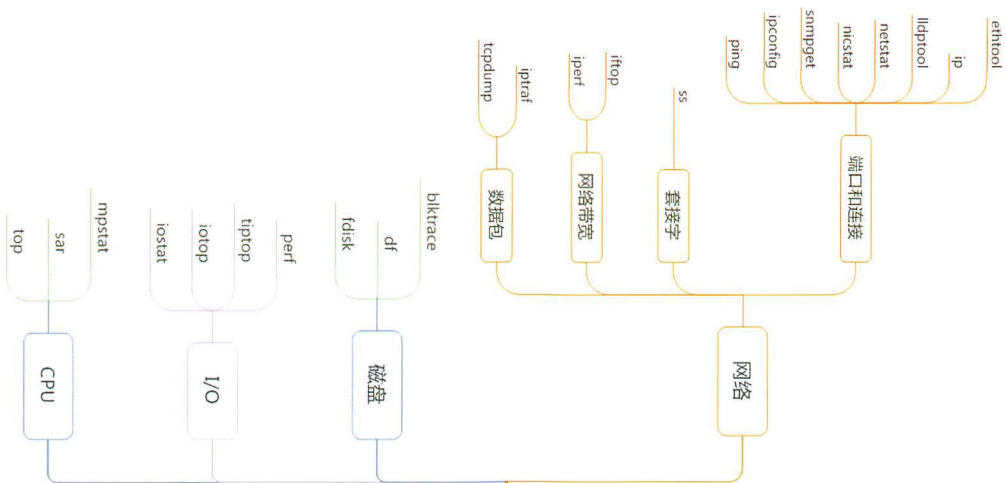
安全测试工具

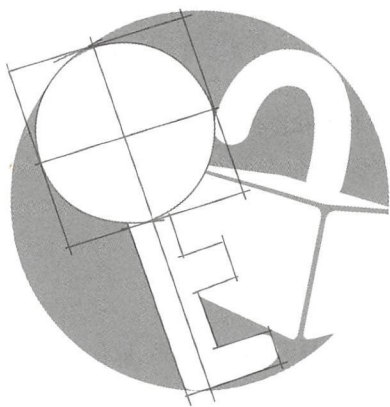
- 漏洞扫描工具
 - Nessus
 - Metasploit
 - OpenVAS
 - SecWiki
- 代码测试工具
 - JUnit
 - TestNG
 - JUnit4
 - JUnit5
- 性能测试工具
 - JMeter
 - LoadRunner
 - WebLoad
 - QTP
- 安全测试工具
 - Wireshark
 - SniffTool
 - Sniffer
 - Snort











质量全面管控

从项目管理到容灾测试

葛长芝 鲁盈盈 欧仕强 编著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书专门为有志于软件测试的工程师打开软件测试的大门,笔者结合案例讲解测试实践技术,主要内容有:项目管理、项目需求管理、代码质量控制、自动化部署、软件测试、安全测试与安全管理、自动化测试基础、自动化测试框架、性能测试、性能分析、监控平台与故障排查、灾难恢复与容灾测试等。书中使用了大量的原创图表,提供了基础工具的使用方法和流程。

本书图文并茂,通俗易懂,提供的大量实例可以使读者边学习边实践,深入理解书中的内容,并将所学到的知识应用于实际项目中,对于初中级软件测试工程师来说是不可多得的工具书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

质量全面管控:从项目管理到容灾测试 / 葛长芝, 鲁盈盈, 欧仕强编著. —北京: 电子工业出版社, 2017.3
ISBN 978-7-121-30786-7

I. ①质… II. ①葛… ②鲁… ③欧… III. ①软件—测试—基本知识 IV. ①TP311.55

中国版本图书馆 CIP 数据核字 (2017) 第 004558 号

策划编辑: 孙学瑛

责任编辑: 徐津平

印 刷: 三河市双峰印刷装订有限公司

装 订: 三河市双峰印刷装订有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×980 1/16 印张: 31.5 字数: 614 千字 彩插: 4

版 次: 2017 年 3 月第 1 版

印 次: 2017 年 3 月第 1 次印刷

定 价: 79.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zlbs@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: (010) 51260888-819, faq@phei.com.cn。

前 言

本书主要内容

写作本书的目的是为软件测试人员提供指导方向，笔者结合案例讲解测试实践技术，非常适合初中级测试人员阅读。本书内容从项目管理、需求管理开始，到各类测试方法的讲解。读者可以按顺序阅读，也可以选择其中的某几章来阅读。本书共有 12 章，每章的主要内容如下：

- 第 1 章：项目管理，主要介绍软件开发过程中的项目管理，包含项目管理的流程、输入输出项及相关工具，不论你是项目管理人员，还是研发人员，都需要了解项目，并通过这些项目管理工具来有效地管理项目。
- 第 2 章：项目需求管理，主要介绍软件需求的定义和流程、软件质量与需求的关系、在编写软件需求过程中应该注意的问题、如何发现软件需求中的问题，以及如何处理需求变更。
- 第 3 章：代码质量控制，主要介绍如何有效地保证代码质量，通过构建 SonarQube 发现代码质量问题，对不规范的代码提出建议并规范。
- 第 4 章：自动化部署，主要介绍部署配置中心和一键部署系统、使用 JDeploy 自动化部署代码、采用 Jenkins 进行持续构建来完成自动化部署任务，从而使部署系统简单化和标准化。
- 第 5 章：软件测试，主要介绍功能测试，包含功能测试流程、根据需求原型设计测试用例、发现缺陷并描述这些缺陷，以及功能测试中常见的误区和经验。

- 第 6 章：安全测试与安全管理，主要介绍安全测试的流程、如何发现漏洞和这些漏洞对系统进行的攻击、对攻击进行的防御措施，以及安全管理过程中的开发安全规范和安全管理平台 OSSIM。
- 第 7 章：自动化测试基础，主要介绍自动化测试技术，包含 Web 自动化测试、接口自动化测试和 MOCK 测试，以及开源自动化测试工具 Selenium 和 TestNG 等。
- 第 8 章：自动化测试框架，主要介绍自动化测试框架的基础和实践。
- 第 9 章：性能测试，主要介绍性能测试的基础和测试流程，包括 JMeter 和 LoadRunner 的使用、设计性能测试方案和进行性能调优。
- 第 10 章：性能分析，主要介绍对 JVM 和系统资源进行监控，并分析测试结果。
- 第 11 章：监控平台与故障排查，主要介绍 Zabbix 和 Grafana，以及如何排查服务器故障。
- 第 12 章：灾难恢复与容灾测试，主要介绍容灾的概念，包含容灾的方法、手段、目标、策略和远期规划，以及容灾测试的误区。

本书适合的读者

本书图文并茂，通俗易懂，书中提供了大量的实例，读者可以边学习边实践，深入理解书中的内容，并将所学到的知识应用于实际项目中。

对测试感兴趣的读者，可以从本书中找到各种测试类型的介绍，包括功能测试、安全测试、自动化测试、性能测试和容灾测试。对于项目管理、需求管理和监控感兴趣的读者，也可以从本书中获益。

致谢

真诚感谢笔者团队成员的参与，群策群力、互相帮助终于完成此书，发自内心地希望读者可以从中得到启发，并且能把一些实战经验应用于具体的项目中。

感谢家人，写作占用了大量的时间和精力，无暇顾及家庭，所以那句老话永远是正确的，军功章有我的一半也有你的一半。

由于作者水平有限，书中不足及错误之处在所难免，敬请专家和读者批评指正。
为了与读者可以及时沟通和收集反馈意见，本书作者特此留下 QQ: 34334546。

葛长芝

2016 年 12 月

轻松注册成为博文视点社区用户（www.broadview.com.cn），您即可享受以下服务：

- **提交勘误：**您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与作者交流：**在页面下方【读者评论】处留下您的疑问或观点，与作者和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/30786>

二维码：



目 录

第 1 章 项目管理	1
1.1 项目管理概述	2
1.2 软件项目管理	5
1.3 项目质量管理	10
1.4 项目管理流程	11
1.4.1 项目启动	11
1.4.2 项目计划	13
1.4.3 软件需求阶段	14
1.4.4 变更管理	16
1.4.5 设计阶段	18
1.4.6 构建阶段	19
1.4.7 测试阶段	21
1.4.8 部署与试运行	23
1.4.9 项目总结	25
1.5 项目管理十诫	26
1.6 项目管理工具对比	28
1.6.1 Microsoft Project	28
1.6.2 Redmine	29
1.6.3 Feng Office	30

1.6.4	ProjQtOr	32
1.6.5	Project-Open	33
1.6.6	各管理工具特性对比	33
1.7	ProjQtOr 简介	34
1.7.1	部署 ProjQtOr	34
1.7.2	常用术语	38
1.7.3	系统配置—系统管理员	39
1.7.4	需求管理—产品经理	43
1.7.5	项目管理—项目经理	45
1.7.6	开发管理—开发人员	48
1.7.7	测试管理—测试人员	50
1.8	要点回顾	52
第 2 章	项目需求管理	53
2.1	认识需求	53
2.1.1	需求的基本定义	54
2.1.2	需求类别	54
2.1.3	需求干系人	56
2.2	需求工程	57
2.3	需求开发	57
2.3.1	需求调研	58
2.3.2	需求分析	61
2.3.3	需求定义	64
2.4	需求管理	66
2.4.1	需求确认	66
2.4.2	需求跟踪	67
2.4.3	需求变更管理	68
2.5	需求说明书	69
2.5.1	《业务需求规格说明书》	70
2.5.2	《用户需求规格说明书》	71

2.5.3	《软件需求规格说明书》	72
2.6	测试需求	73
2.7	需求管理工具	74
2.7.1	Rational RequisitePro	75
2.7.2	TelelogicDoors	76
2.8	Plandora 实战	76
2.8.1	搭建 Plandora	77
2.8.2	管理员配置	78
2.8.3	前端用户	82
2.8.4	需求变更实例	88
2.9	要点回顾	92
第 3 章	代码质量控制	93
3.1	静态代码分析	94
3.2	代码文档规范	94
3.2.1	命名规范	95
3.2.2	编码规范	96
3.2.3	注释规范	97
3.2.4	异常处理规范	102
3.2.5	单元测试规范	103
3.2.6	文件解析规范	103
3.3	Sonar 简介	106
3.3.1	Sonar 体系架构	107
3.3.2	Sonar 代码规则	108
3.4	Sonar 服务端	110
3.4.1	环境要求	110
3.4.2	Sonar 服务器搭建	110
3.4.3	Sonar 配置	112
3.4.4	Sonar 插件	114
3.5	Sonar 客户端	116

3.5.1	Sonar-Runner 客户端	116
3.5.2	Maven 插件方式	117
3.5.3	Ant 插件方式	118
3.5.4	Eclipse 插件方式	119
3.6	最佳实践	121
3.6.1	项目配置	122
3.6.2	持续审查	123
3.6.3	结果分析	125
3.6.4	集成曲线图	128
3.7	要点回顾	129
第 4 章	自动化部署	130
4.1	引入自动化部署	131
4.1.1	复杂的手工部署	131
4.1.2	自动化部署方式	132
4.1.3	持续集成思想	133
4.2	自动化部署的特点	134
4.2.1	环境一致性	134
4.2.2	部署系统化	134
4.2.3	配置集中化	135
4.3	版本控制	137
4.3.1	Git 简介	137
4.3.2	Git 部署	138
4.3.3	Git 客户端使用	141
4.3.4	Git 相关操作	144
4.3.5	代码管理	147
4.4	JDeploy 平台	151
4.4.1	认识 JDeploy	151
4.4.2	JDeploy 部署配置	151
4.4.3	一键部署项目	152

4.5	要点回顾	155
第5章	软件测试	156
5.1	软件测试	156
5.1.1	软件测试发展史	157
5.1.2	软件测试的未来	160
5.1.3	测试部门组织架构	161
5.1.4	软件测试的基本类型	163
5.2	测试模型	164
5.2.1	瀑布模型	164
5.2.2	敏捷模型	165
5.2.3	敏捷测试与传统测试的区别	166
5.3	系统测试流程	167
5.4	根据需求原型设计测试用例	171
5.4.1	需求原型规范样式	171
5.4.2	设计测试用例	173
5.5	缺陷描述	175
5.5.1	缺陷属性	175
5.5.2	缺陷描述示例	176
5.6	测试的策略	178
5.7	测试过程的经验总结	179
5.8	质量保证	180
5.9	要点回顾	181
第6章	安全测试与安全管理	182
6.1	安全测试	182
6.1.1	安全测试概述	183
6.1.2	安全测试的基本过程	184
6.1.3	安全测试与安全运维	186
6.1.4	安全测试工具	186

6.1.5	安全测试用例	187
6.2	开发安全规范	189
6.2.1	跨站脚本安全规范	189
6.2.2	SQL 注入安全规范	191
6.2.3	页面组件和敏感数据的安全规范	193
6.2.4	Java 安全规范	196
6.2.5	应用集成安全规范	197
6.3	代码安全审核工具 Checkmarx	197
6.3.1	SQL 注入	198
6.3.2	反射型跨站脚本攻击	199
6.3.3	储存型 XSS	201
6.3.4	HTTP 响应头分裂 (Http_Response_Splitting)	201
6.4	安全漏洞	202
6.4.1	信息收集	202
6.4.2	口令入侵	204
6.4.3	心脏滴血漏洞	207
6.5	OSSIM 安全管理平台	209
6.5.1	OSSIM 架构	210
6.5.2	安装与部署	211
6.5.3	OSSIM 控制台	211
6.5.4	Web 界面配置	212
6.5.5	OSSIM 使用实战	214
6.6	要点回顾	219
第 7 章	自动化测试基础	220
7.1	自动化基础	221
7.1.1	自动化测试流程	222
7.1.2	自动化测试特点	225
7.1.3	自动化测试工具	226
7.1.4	标记语言介绍	228

7.2	Web 自动化测试	233
7.2.1	元素定位	234
7.2.2	Selenium IDE	238
7.2.3	Selenium 使用	241
7.3	接口自动化测试	247
7.3.1	接口测试类型	248
7.3.2	接口测试工具	249
7.3.3	Mock 测试	250
7.3.4	HTTP 协议测试	254
7.4	TestNG 框架	258
7.4.1	TestNG 配置	258
7.4.2	TestNG 注解	260
7.4.3	测试套件	260
7.4.4	数据驱动	265
7.4.5	执行测试结果	267
7.4.6	测试集成	268
7.5	要点回顾	270
第 8 章	自动化测试框架	271
8.1	框架分析	272
8.1.1	框架设计目标	272
8.1.2	业务流程层次分析	272
8.1.3	业务流程测试自动化	273
8.1.4	手工用例自动化	274
8.2	框架设计	274
8.2.1	框架设计思想	274
8.2.2	框架物理架构	276
8.2.3	框架逻辑架构	277
8.2.4	框架工作流程	278
8.3	框架开发	278

8.3.1	创建测试用例	279
8.3.2	创建测试数据	280
8.3.3	创建测试项目	281
8.3.4	开发框架运行类	285
8.3.5	开发公共接口	288
8.3.6	添加日志报告	292
8.4	脚本开发	292
8.4.1	编写测试脚本	292
8.4.2	调试运行脚本	293
8.4.3	上传脚本	294
8.5	持续集成	296
8.5.1	Jenkins 服务器搭建	296
8.5.2	Jenkins 相关插件	298
8.5.3	部署测试执行机	299
8.5.4	分布式测试	300
8.5.5	配置测试任务	303
8.5.6	查看运行结果	310
8.6	要点回顾	312
第 9 章	性能测试	313
9.1	性能测试基础	313
9.1.1	性能术语	314
9.1.2	需求分析与策略	317
9.2	测试利器之 LoadRunner	321
9.2.1	LoadRunner 安装贴士	321
9.2.2	脚本与优化	322
9.2.3	设置场景	339
9.2.4	运行场景	340
9.2.5	收集和分析结果	341
9.3	测试利器之 JMeter	343

9.3.1	JMeter 介绍	343
9.3.2	JMeter 脚本与优化	344
9.3.3	收集监控数据	353
9.3.4	运行测试	353
9.3.5	JMeter 使用小结	354
9.4	性能测试框架搭建	355
9.4.1	JMeter 配置监听器	355
9.4.2	InfluxDB 数据库配置	356
9.4.3	InfluxDB Graphite Listener 配置	357
9.4.4	查看 InfluxDB 结果	357
9.4.5	Grafana 配置	358
9.5	性能测试实战	359
9.5.1	明确测试需求	359
9.5.2	选取测试方法和策略	361
9.5.3	准备测试脚本	362
9.5.4	执行与分析测试结果	362
9.5.5	提出调优建议	364
9.5.6	交付测试报告	364
9.6	性能调优	364
9.6.1	CPU 使用率过高	364
9.6.2	I/O 使用率过高	365
9.6.3	进程数调整	367
9.6.4	线程不安全	367
9.6.5	数据库连接数过少	368
9.6.6	数据导入慢	369
9.7	要点回顾	369
第 10 章	性能分析	371
10.1	系统硬件资源监控	372
10.1.1	nmon 工具	373

10.1.2 Linux 系统监控命令	378
10.2 JVM 监控与分析	386
10.2.1 JVM 基础	386
10.2.2 JVM 垃圾回收	388
10.2.3 常见 JVM 命令	390
10.2.4 堆分析工具 MAT	402
10.2.5 JConsole	403
10.2.6 JProfiler	406
10.3 数据库性能分析	415
10.3.1 软解析和硬解析	415
10.3.2 SQL 执行计划分析	417
10.3.3 数据库连接数监控	418
10.3.4 Oracle 数据库性能诊断报告 AWR	419
10.4 要点回顾	427
第 11 章 监控平台与故障排查	428
11.1 监控系统	428
11.1.1 日志监控平台	429
11.1.2 硬件和应用监控平台	429
11.2 Zabbix 简介	430
11.2.1 系统架构	430
11.2.2 配置 Zabbix	432
11.2.3 常见的配置问题	436
11.2.4 监控主机	438
11.3 美化界面 Grafana	441
11.3.1 部署 Grafana	441
11.3.2 使用 Grafana	442
11.4 服务器故障排查	447
11.4.1 清楚故障的前因后果	447
11.4.2 搜寻蛛丝马迹	447

11.4.3	列出当前运行的进程	449
11.4.4	监听网络服务	451
11.4.5	查看硬件状态	451
11.4.6	列出挂载点和文件系统	457
11.4.7	过滤内核和中断信息	459
11.4.8	定时任务	460
11.4.9	分析系统日志	461
11.5	要点回顾	461
第 12 章	灾难恢复与容灾测试	462
12.1	灾难恢复	463
12.1.1	灾难恢复的规范	463
12.1.2	灾难恢复能力等级	466
12.1.3	灾难恢复的关键指标	468
12.2	容灾测试	471
12.2.1	容灾的起源	471
12.2.2	容灾的定义	472
12.2.3	容灾的区别	474
12.3	详解容灾测试	475
12.3.1	容灾测试的目标	475
12.3.2	职责的划分	476
12.3.3	容灾测试的流程	477
12.4	容灾测试实战	478
12.4.1	容灾测试计划	479
12.4.2	容灾用例与 Bug	480
12.4.3	容灾线上演习	484
12.4.4	容灾长期规划	485
参考文献		486

第 1 章

项目管理

要了解项目管理，首先要弄清楚什么是项目，以及项目具有哪些特性，只有做到未雨绸缪，才能按部就班地针对这些特性进行有效的管理。

项目是为创造独特的产品、服务或者成果而进行的临时性工作。

项目具有三大特性：独特性、临时性和不确定性。

- 独特性：产品独特，预先不可见，很难重复，不能完全照搬。
- 临时性：有明确的开始时间和结束时间，项目组是临时组成的，流动性强。
- 不确定性：肯定存在风险和意外，需要进行风险控制。

提示：笔者提到的项目特性不是 PMP 书里面所讲的内容，都是项目经验的累积。因为读者阅读本书的目的不是参加 PMP 考试，所以笔者把自己的经验分享给读者。当然，经验有很大的局限性，因为公司不同、项目不同、背景不同，可能和读者的认知产生分歧，所以经验只是参考。

项目既然具有独特性、临时性和不确定性，那么相当多的问题就会自然而然地出现，因此必须引入项目管理，否则项目管理工作就会一团乱麻、无从着手。

项目管理就是将管理的知识、工具和技术用于项目活动上，解决项目的问题，从而实现项目的需求。项目管理中重要的是一定要在范围、时间、成本和质量这些相互间有冲突的因素中寻求平衡，实现有不同需求和期望的干系人的目标。这里的项目干系人是指积极参与项目、其利益会受到项目执行或完成情况影响的个人或组织。如图 1-1 所示非常清晰地展示了项目管理的本质。

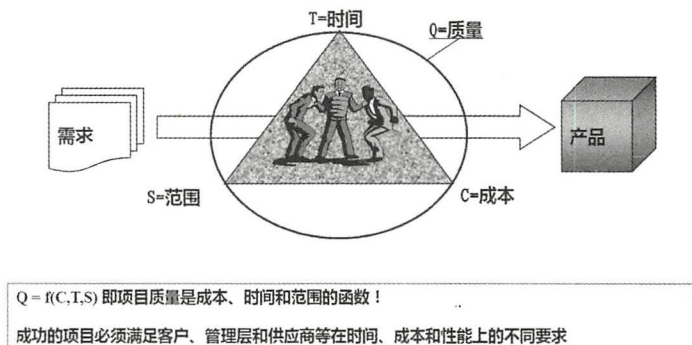


图 1-1 项目管理的本质

通过阅读本章，读者可以了解以下内容：

- 项目与项目管理；
- 项目管理和质量管理之间的关系；
- 项目管理的整个流程；
- 项目各个阶段的流程及输入输出条件；
- 决定项目成功的关键因素；
- 常用的项目管理工具；
- 使用 ProjeQtOr 来管理项目。

1.1 项目管理概述

因为本章的核心是讲解项目管理，所以要把项目管理延展开进行详细说明，由起源、发展和完善，再扩展到国内项目管理的应用情况，分别进行阐述。

传统的项目与项目管理起源于建筑业和建筑工程管理，并且有着悠久的实践历史，例如中国的长城、埃及的金字塔等。当代项目与项目管理开始于大型国防工业，例如美国的北斗星导弹项目。国际项目管理学术组织的出现标志着项目管理走向了科学，国际项目管理协会（IPMA）成立于 1965 年，美国项目管理学会（PMI）成立于 1969 年，这两大组织的成立标志着项目管理正式成为一门学科、一个专业、一种职业。

通过表 1-1 读者能够清楚地了解项目管理的发展史。在 2000 年之前，项目管理的发展



速度比较均匀,各种概念或者模型都经历长时间的验证才形成一套理论。在2000年之后,项目管理处于快速发展阶段。这其中有三个方面的原因,第一个是经济原因,因为经济的飞速发展,各行各业制定了自己的管理方法;第二个原因是软件公司的大量出现,这个对精细度要求非常高的行业产生了很多的管理标准;第三个原因毋庸置疑,一流的企业卖标准,二流的企业卖品牌,三流的企业卖产品,四流的企业卖苦力,所以各个大企业开始推销自己的管理标准。

表 1-1 项目管理的发展史

20 世纪初	Henry Gantt 发明甘特图
20 世纪 50 年代	出现 4 种计划方法: PERT、CPM、ADM 和 PDM
20 世纪 50 年代	出现单点责任概念
1960 年	NASA 开发出矩阵型组织结构
1963 年	美国空军开发出挣值管理方法
1964 年	NASA 提出配置管理
1965 年	国际项目管理协会 (IPMA) 成立
1969 年	美国项目管理学会 (PMI) 成立
1984 年	PMI 推出 PMP 考试
1999 年	我国开始 PMP 考试
2000 年	摩托罗拉中国软件中心通过 CMM 第五级认证
2001 年	美国犹他州的雪鸟城成立敏捷软件开发联盟
2010 年	国内的企业通过 CMMI, 工程师考取证书的热情锐减

国内的企业十分热衷于讲述自己的管理制度,并参加国际上的各种管理认证。于是在短短的十年间,我们被各种管理制度、管理体系、管理规范和管理证书所包围。各种各样的培训机构、西装革履的培训讲师、激扬奋进的演讲视频冲击着我们的的大脑神经,在机场、商场、会展和火车站,即使一闪而过,标志性的装扮和宏亮的声音也会瞬间让我们知道这是管理大师、营销专家。如果驻足一段时间,仔细倾听,我们能够记住的差不多只有一些插科打诨的段子,至于讲解的管理知识是很难再次想起的。这其中最关键的有两点:一是理论太多,条条框框,根本无法实践到工作中;二是这些管理成功学往往例举的都是小概率的成功事件,根本没有样本性。这些管理学培训忽略了个体的差异性,忽略了最关键的因素,项目管理最关键的是项目组成员的团结协作能力。

现在的项目与项目管理是已经扩展的广义概念,项目管理更加面向市场和竞争,注重



人的因素、注重顾客、注重柔性管理。项目管理发展的趋势是全球化、多元化和专业化，随着科学技术的飞速发展，产品的规模越来越庞大，将遍布于各个行业，如建筑业、零售业、IT 行业和军工行业。

笔者通过系统的归纳，认为项目就是项目管理者有效地分配资源，对项目中的任务进度进行跟踪管理以完成客户的需求。稍微扩展一下，在有限的资源约束下，运用系统的模型、方法和理论，对项目涉及的全部工作进行有效的管理，即从项目的立项决策开始，到项目结束的全过程，进行计划、组织、协调、控制和评价，以实现项目的目标。

项目管理中的几个重要元素是：项目、任务、资源、计划和控制。这几个元素之间相互关联，也相互影响。项目需要拆分成任务，任务需要分配资源，资源需要进行计划和控制。

我国的项目管理领域有大量来自于国外的管理模型、管理理论、管理证书、方法论和项目管理工具，让人眼花缭乱，如 CMM/CMMI、ISO、IPD、PMP、IPMP、PRINCE2、CPMP、IMCP 和 SCRUM，其中对于企业有两种认证是特别常见的，即六西格玛和 ISO 标准，它们适用于很多行业。表 1-2 对六西格玛和 ISO9000 系列标准进行了简单对比。ISO 最常见的有三个认证，即 ISO20000-IT 服务管理体系认证、ISO27000 信息安全管理体系认证和 ISO9000 质量管理体系认证。

表 1-2 六西格玛和 ISO9000 系列标准对比

	六西格玛	ISO9000 系列标准
起源	1986 年，美国摩托罗拉公司的比尔·史密斯提出六西格玛概念，并在摩托罗拉公司和通用电气公司成功推广	ISO 总部设于瑞士日内瓦，成员包括 162 个会员国，1987 年公布 ISO9000 认证标准
目标	百万分之 3.4 的缺陷率	定义客户满意的关键要素，定义质量的责任，确保质量系统的正确性和有效性，确保每一份合同都满足客户要求
费用	看推广的程度	根据培训机构的不同而不同
难度	十分困难	极其简单
级别	绿带、黑带、大师级黑带、冠军	没有级别
方法	DMAIC 方法和 DMADV 方法	PDCA 循环，也叫戴明循环
过程	分析软件，项目管理软件，数据整理软件，项目协调软件	成立文件编写小组； 接受质量体系培训； 编写质量体系文件； 质量标准试运行； 认证机构组织模拟审核



续表

	六西格玛	ISO9000 系列标准
核心	品质控制、全面品质管理、零缺陷法	强化品质管理, 消除国际贸易壁垒, 节省审核精力和费用, 稳定企业内部经营运作
原则	对获得可测量、可量化的财务回报有明确的规定; 由冠军、大师级黑带、黑带等来领导和贯彻六西格玛方法的执行	八项品质管理原则: 以顾客为关注焦点; 领导作用; 全员参与; 过程方法; 管理的系统方法; 持续改进; 基于事实的决策方法; 与供方互利的关系
证书	个人: 六西格玛绿带和六西格玛黑带 企业: 财务回报	个人: 内审员 企业: 质量管理体系认证证书

提示: 国际标准化组织 (ISO) 2007 年 11 月发布的调查结果显示, 截至 2006 年年底, 在 170 个国家共颁发了 ISO9001: 2000 版认证证书 897 866 张, 其中在中国颁发了 162 259 张证书, 占颁发总量的 18%, 居世界第一位。现在稍微有点规模的企业, 都已经通过了或者正在准备通过 ISO9000 标准认证, 但它只是对企业实施最佳标准管理的认证。对消费者而言, 用 ISO9000 标准来评判产品的质量是没有说服力的。不过, 在企业看来, 拿到这个认证, 能够给消费者传达一个很积极的信号。而六西格玛是没有证书的, 由实施项目的财务回报作为评判成功还是失败的标准。

提示: 计算机企业热衷于 CMM/CMMI 认证是有多方面原因的, 第一个是趋势, 在 2000 年摩托罗拉公司通过 CMM5 认证后, 国内的软件公司陆续通过 CMM/CMMI 认证, 因为当时我国很多软件公司的主要业务是承接国外公司的软件外包, 而国外公司选择软件开发商的主要依据就是是否通过了 CMM/CMMI 认证; 第二个是政府的补贴, 软件公司大多数都是在高新区注册的, 通过各种资质认证, 政府会提供相关补贴。

1.2 软件项目管理

从概念上讲, 软件项目管理是为了使软件项目能够按照预定的成本、进度和质量顺利



质量全面管控：从项目管理到容灾测试

完成，而对成本、人员、进度、质量和风险等进行分析和管理的活动。软件开发不同于其他产品的制造，软件开发的整个过程都是设计过程（没有制造过程）。另外，软件开发不需要使用大量的物质资源，主要是人力资源，并且软件开发的产品只是程序代码和技术文件，并没有其他的物质结果。基于上述特点，软件项目管理与其他项目管理相比，有很大的独特性。下面是软件项目管理的几个属性。

- 一个目标：实现项目干系人对项目的要求和期望。
- 两个管理：项目干系人管理，项目干系人的需求与期望不断的变化；项目阶段管理，项目产生和存在的环境也在不断变化。
- 三种约束：时间、成本和范围，任何一个元素的变化都会引起整体的变化。
- 四个阶段：定义（需要做什么）、计划（什么时候做）、实施（如何做）和收尾（验证是否正确）。
- 五个过程：启动、计划、执行、监控和收尾。
- 九大体系：整合、范围、时间、成本、质量、人力、沟通、风险和采购。

以上属性是从 PMP 中梳理出来的，它对应的是传统的 CMMI 管理模型，共分为 5 个级别、22 个 KPA（Key Process Area）。PMP 管理理论和 CMMI 管理模型都采用传统瀑布式软件开发模型，如图 1-2 所示。

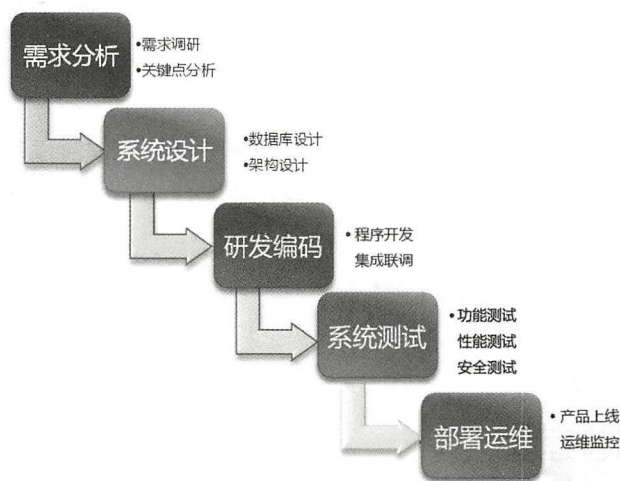


图 1-2 瀑布式软件开发模型



软件项目开发过程除了传统的瀑布型开发（Waterfall Development）模式，还有敏捷开发（Agile Development）模式。敏捷开发模式只是一个笼统的概念，可以细分出很多子模式。

- 极限编程（Extreme Programming，简称 XP）。XP 是由 KentBeck 在 1996 年提出的，是一种近螺旋式的开发方法。XP 注重的核心是沟通、简明、反馈和勇气，要求纪律性，提倡测试先行。
- Scrum。Ken Schwaber 和 Jeff Sutherland 于 1995 年 OOPSLA 大会上首次介绍了 Scrum。Scrum 是一个包括了一系列实践和预定义角色的过程，是迭代的增量化过程。
- 透明水晶方法（Crystal）。Crystal 是由 Alistair Cockburn 和 Jim Highsmith 建立的敏捷方法系列，其目的是发展一种提倡“机动性的”方法。
- 特征驱动开发（Feature-Driven Development，简称 FDD）。FDD 由 Peter Coad、Jeff de Luca 和 Eric Lefebvre 共同开发，1999 年首次在《Java 彩色 UML 建模》书中提出，是一个以客户为中心和以架构为中心的实用软件过程。
- 自适应软件开发（Adaptive Software Development，简称 ASD）。ASD 由 Jim Highsmith 在 1999 年正式提出。ASD 关注于快速创建开发软件系统。

如图 1-3 所示是典型的敏捷开发模式的项目开发过程。从需求、计划、开发、测试，直到回顾，整个周期一直在迭代中，而其中的开发、测试、发布又可以单独迭代多次。

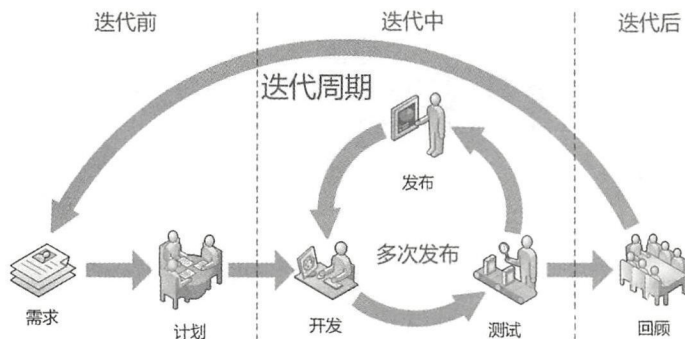


图 1-3 敏捷开发过程

采用敏捷开发模式容易有一些误区。

- 误区一：敏捷开发意味着不要文档。敏捷开发同样需要文档，敏捷只是一些实践



质量全面管控：从项目管理到容灾测试

的结合。

- 误区二：敏捷只适用于小项目。
- 误区三：敏捷只会对研发产生改变，管理者不需要亲自了解敏捷，只要管理上支持。
- 误区四：敏捷只注重快速交付，架构不重要。

有适合传统瀑布式开发的 CMMI 管理模型，也有适合敏捷开发的 Scrum 模型。图 1-4 很好地解释了瀑布式模型与敏捷式模型的差别。瀑布式模型分为计划、分析、设计、程序开发、测试、修改和整合，一个阶段结束，另一个阶段接着开始。而敏捷式模型着重迭代式开发，分析、设计、开发、测试和发布会不停地迭代直至需求完成、产品上线。

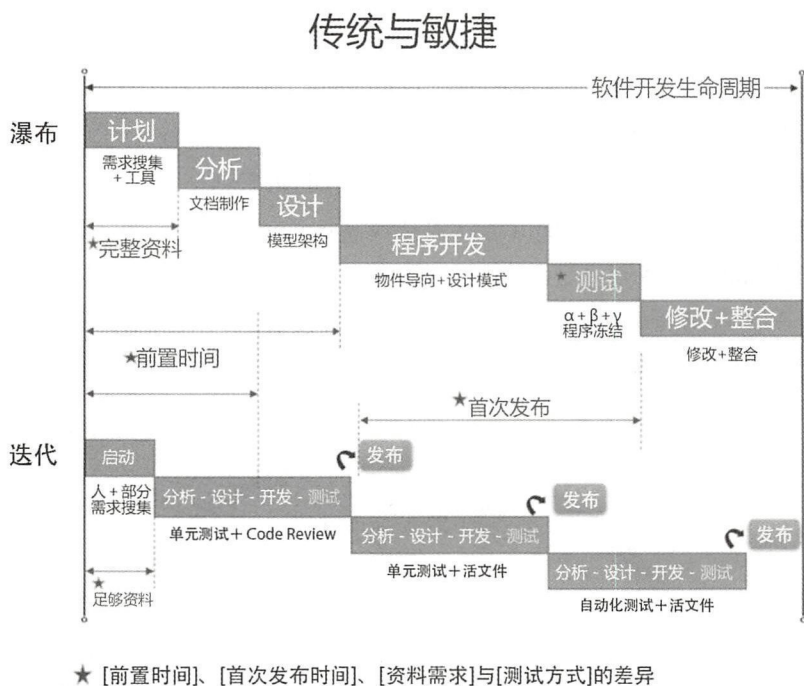


图 1-4 瀑布式模型与敏捷式模型的区别

选择项目开发模式时，读者应该明白没有哪个模型是绝对正确的。正如英国的乔治·博克斯（George Box）说过的一句话，“所有的模型都是错的，但有些却是有用的”。选择模型时，一定要选择适合这个项目的，要在实践过程中不停地调整，使其适应整个项目开发，





并且保证执行顺利。

提示：适合公司的、项目的、团队的模型就是最好的模型。

上面主要讲解了公司的管理规范 and 开发模式，下面讲解软件工程师关注的管理证书。

针对个人项目管理证书的管理体系如表 1-3 所示。

表 1-3 项目管理体系的对比

	PRINCE2 受控环境下的项目	IPMP 国际项目管理资质认证	PMP 项目管理专业人员
颁证机构	英国政府商务部（OGC）	国际项目管理协会（IPMA）	项目管理协会（PMI）
官方标准	英国、澳大利亚和联合国标准	欧洲	美国
推广组织	APMG 和 ATO	IPMA 和认证考点	PMI 和 REP
推广范围	150 多个国家，2008 年进入中国	36 个国家，2001 年进入中国	130 多个国家，1999 年进入中国
认证体系	两级认证体系（报考不受资历与水平限制）： 基础资格（Foundation）； 从业资格（Practitioner）	四级认证体系（有资历和水平等级区别）： A 级，国际特级项目经理； B 级，国际高级项目经理； C 级，国际项目经理； D 级，国际助理项目经理	单一证书制，一个级别（没有资历和水平区别），PMP 项目管理专家
考核方式	均为笔试（题型：选择题）	报告/案例+笔试+面试	机考（题型：选择题）
认证标准	PRINCE2	ICB	PMBOK
证书有效期	基础资格终身有效，从业资格 5 年有效，必须逐级申请	A、B、C 级 5 年有效，D 级终身有效，可依据个人资历申请对应的级别	3 年有效，3 年内 60 个 PDU 专业学习
体系结构	（1）七大原则：持续的商业论证、参考以前的经验、定义角色和责任、分阶段管理、例外管理、重点关注产品、根据项目环境增减。 （2）七大主题：商业论证、组织、控制、风险、质量、变更、进展。 （3）七大过程：项目准备、项目启动、项目指导、阶段控制、阶段边界管理、产品交付管理、项目收尾	（1）项目管理知识体系： • 项目和项目管理； • 项目开发的四个阶段，即概念阶段、规划阶段、实施阶段和收尾阶段； • 项目管理方法与工具。 （2）ICB3.0 能力基准	（1）项目及生命周期：项目定义、项目管理定义、项目生命周期定义。 （2）五大过程组：启动、规划、监控、执行、收尾。 （3）九大知识领域：整合、范围、成本、进度、质量、人力资源、沟通、风险、采购

PRINCE2、PMP 和 IPMP 三大国际主流项目管理体系是目前行业内最流行的。IT 工程师或者 IT 项目经理非常热衷考 PMP 证书有两个原因，第一个原因是 PMP 宣传得当，第二





质量全面管控：从项目管理到容灾测试

个原因是 PMP 证书的推广遥遥领先另外两个证书的推广。PMP 理论通俗易懂，而且在项目上比较容易推广，适合中国的 IT 公司。

1.3 项目质量管理

项目质量管理的过程和活动都是为了满足和达到项目要求的质量。那么究竟什么是质量呢？质量就是产品满足客户的需求，达到预期的目标和性能。ISO 把质量定义为“一个主体的所有特性能够满足已声明或默认的需求”。

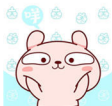
项目质量管理中比较重要的是软件质量保证。软件质量保证（Software Quality Assurance, 简称为 SQA）是一个确保开发的软件产品符合预设的质量需求标准的过程。SQA 伴随着整个软件开发生命周期，可以周期性地检查软件以保证开发出高质量的产品。SQA 可以在多种开发模式下实现。从广义上讲，SQA 也可结合多种测试方法来测试软件。通常 SQA 鉴于工业标准来建立软件质量指南和实施策略，这些标准包括 ISO9000 和能力成熟度模型集成（CMMI）。

需要注意的是，质量管理不是独立的，而是持续的过程。不能等到产品开发结束之后才开始检查质量，而是需要在每个开发阶段都进行质量检查直至项目完成。每个阶段必须满足规定的质量标准才能移步至下一个阶段。质量管理是项目管理中不可或缺的一部分，伴随着项目开始直至终止。质量管理也是可重复的过程，持续进行质量评估、过程更新，直至达到要求的质量。

虽然人们大多认为软件的质量很重要，但是相当多的项目团队成员并不懂得如何有效地改善软件质量属性，如正确性、健壮性、可靠性、性能、易用性、安全性、可扩展性、可复用性、兼容性和可移植性等，不会分析当前软件的质量要素是什么，没有把精力集中在改善对经济效益贡献最大的质量要素上面。

有些软件公司没有软件质量管理的措施，开发人员把完成功能当成终极目标。用户在使用软件的过程中发现许多 Bug（软件缺陷），导致开发人员的纠错性维护代价很高。

有些软件公司虽然很重视软件质量，按照 ISO、CMMI 的要求建立了管理规范，但是效果不明显，因为搞不清楚软件测试、技术评审和质量保证的作用和关系，并且不懂得质





量控制，主要靠修补错误的方式提升质量，代价比较高。

很多公司误以为提高软件质量是质量保证人员和测试人员的责任，没有意识到任何开发人员、管理人员都会对质量产生影响，都要对质量负责。另外，质量保证人员的权力比较小，很难推动质量改进措施。

在此建议项目团队成员要理解“软件质量”及常见的软件质量属性，树立全面软件质量管理的理念（模型），制定软件测试、技术评审、质量保证的规范，并使用方便的工具，帮助团队进行软件质量管理，最终高质量地完成整个项目。

1.4 项目管理流程

当一个软件项目没有合理的计划和安排时，每个项目组成员都不知道项目的任务进行到什么阶段，由谁在负责。开发人员不清楚什么时候设计结束，什么时候编码结束，什么时候测试结束，那么自然最后这个产品不能按时交付。测试人员不知道需求的评定和风险，就会评估错时间，影响测试进程。所以，有效地进行项目管理是非常重要的。

每个项目大致都经过调研立项启动、计划审核、需求分析、需求变更、系统设计、构建开发、测试验收、部署试运行和项目总结的不同阶段。各个阶段又涉及到方方面面的过程，有项目估算过程、风险管理过程、项目管理过程、设计开发过程、需求开发过程、同行评审过程、测试过程和部署运行过程等。

如图 1-5 所示展示了整个项目管理的流程，即项目启动→项目计划→软件需求阶段→设计阶段→构建阶段→测试阶段→部署与试运行→项目总结。

1.4.1 项目启动

在图 1-5 所示的项目启动阶段，将项目的目标、规划与任务进行完整的定义和阐述，形成一份完整的项目工作任务书（Statement Of Work，简称为 SOW），作为项目立项的关键产出。

项目启动会是宣导项目重要性的关键节点，必须就项目目标、上线条件、管理权限和干系人列表达成共识。表 1-4 定义了项目启动阶段的条件活动、标准输入和标准输出，并





质量全面管控：从项目管理到容灾测试

且确定了责任人。其中，“入口条件”表示启动阶段的准入标准，“活动”表示启动阶段有哪些工作，“出口条件”表示启动阶段的产出，“验证人”表示对应的产出负责人。

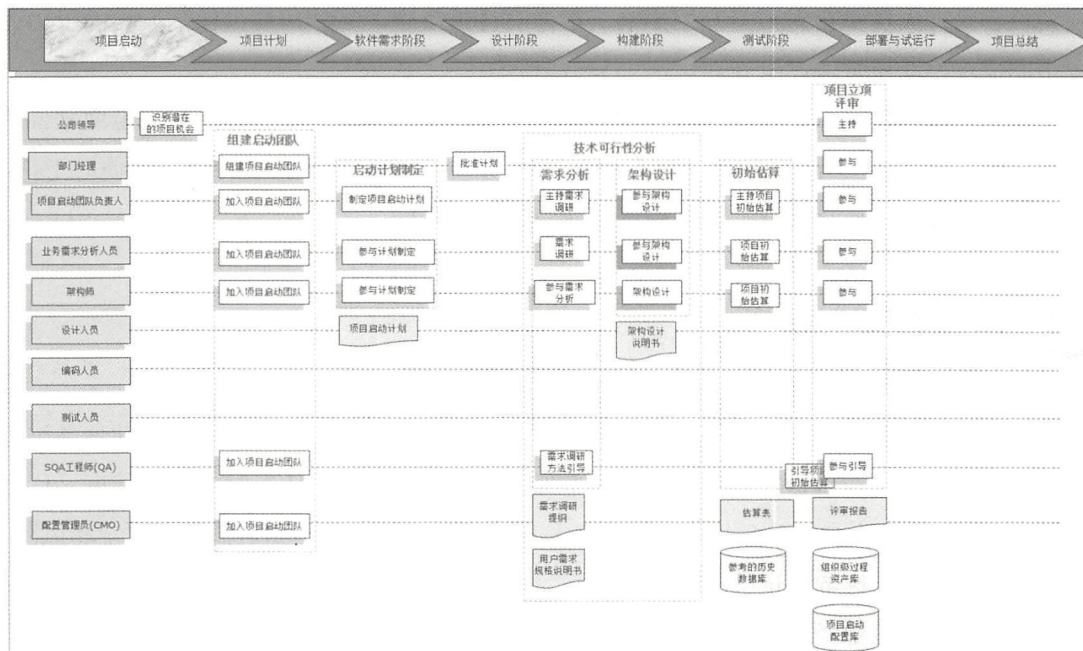


图 1-5 项目管理流程

表 1-4 项目启动阶段的活动和输入输出

入口条件	活动	出口条件	验证人
项目立项会议	(1) 《用户需求说明书》和《架构设计说明书》评审； (2) 《用户需求说明书》和 UI 草图得到客户确认； (3v 完成启动阶段的项目计划和估算； (4) 项目立项申请，通过领导批准； (5) 《用户需求说明书》是否基线化	(1) 《用户需求说明书》和《架构设计说明书》通过评审（附有“通过结论”的评审记录汇总表）； (2) 《用户需求确认单》得到客户签字； (3) 填写完整的《项目启动阶段计划》、《项目估算表》、《规模估算表》； (4) 《项目立项决策分析表》、《项目立项书》填写完整，包括成本、领导批准的记录； (5) 《用户需求说明书》有基线号； (6) 输出 UI 草图	(1) 《用户需求说明书》：开发经理。 (2) 《架构设计说明书》：开发经理、设计师。 (3) 《项目启动阶段计划》、《项目估算表》、《规模估算表》、《项目立项决策分析表》、《项目立项书》：项目经理。 (4) 评审结论、批准记录、客户签字、是否基线化、配置项是否填写：质量保证员。 (5) 配置项归档、基线化：配置管理员。 (6) UI 草图：美工



1.4.2 项目计划

项目启动后,就需要进行合理的计划,包括里程碑和基线时间设定、人员安排和风险预测。首先,进行项目任务书编制和讨论,然后批准下达,之后可以组建开发测试团队,进行项目成本的人力估算、项目计划制定和评审。每个项目组成员根据自己的角色需要进行策略制定与评估,最终得出一份项目计划阶段的审计报告。如图 1-6 所示是计划阶段的各角色和产出。

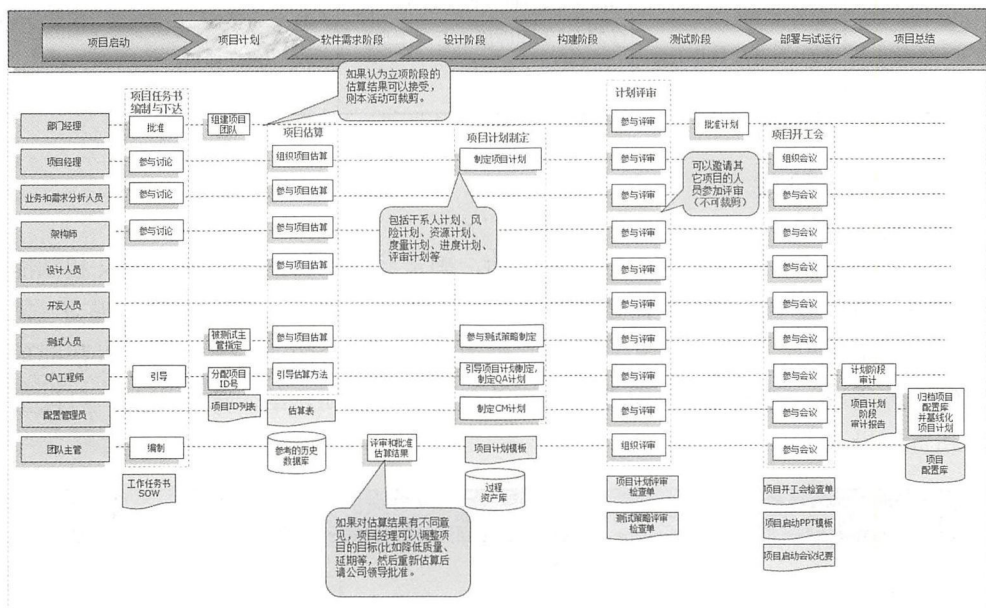


图 1-6 项目计划阶段

读者会有疑问,公司根本没有计划,大家就是口头表述,开个会就立项,针对这样的情况,笔者有如下建议:

- 特别紧急的项目,至少有高层(总经理或者 CTO 级别)知晓;
- 如果有项目管理系统,那么一定要把基本情况全部记录在系统里;
- 如果没有项目管理系统,那么对项目组成员来说,最重要的就是来往的邮件;
- 什么文档都可以没有,需求文档一定要有,这是最基本的条件。

项目经理一生都在做计划,不是计划没有用,而是要针对不同的事件进行调整,这样手握计划才能做到心不慌。例如,项目经理去异地工作需要业务部门之间协调系统上线推广并签字验收,出差前做一份计划,不仅明确出差时间、工作计划,还要把计划传达到

各个业务部门，要求在指定时间内派出人员协助工作，并对工作成果确认签字。如果没有计划，各个业务部门不明白自己应该如何配合，出差任务也就无法按时完成。

表 1-5 项目计划阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 项目批准立项; (2) 《工作任务书》 经过批准和签发	(1) 《项目计划书》 和估算评审; (2) 召开项目开工会	(1) 输出《项目计划书》和估算, 以及通过评审的会议纪要; (2) 《里程碑成本管理手册》 内容填写完整	(1) 《项目计划书》、估算、《里程碑成本管理手册》: 项目经理。 (2) 评审结论、会议纪要、配置项是否填写: 质量保证员。 (3) 配置项归档、基线化: 配置管理员

在软件需求阶段，要分析客户的业务活动，确定系统的目的、范围、定义和功能，明确在用户的业务环境中软件系统需要“做什么”，如图 1-7 所示。需求人员要提交《需求规格说明书》用于评审、估算成本和总结，需求说明书中包含的业务流程图可以帮助项目组成员理解业务需求。测试人员也需要参与需求分析、评审和总结。如有需要，可以重新估算成本和资源投入。

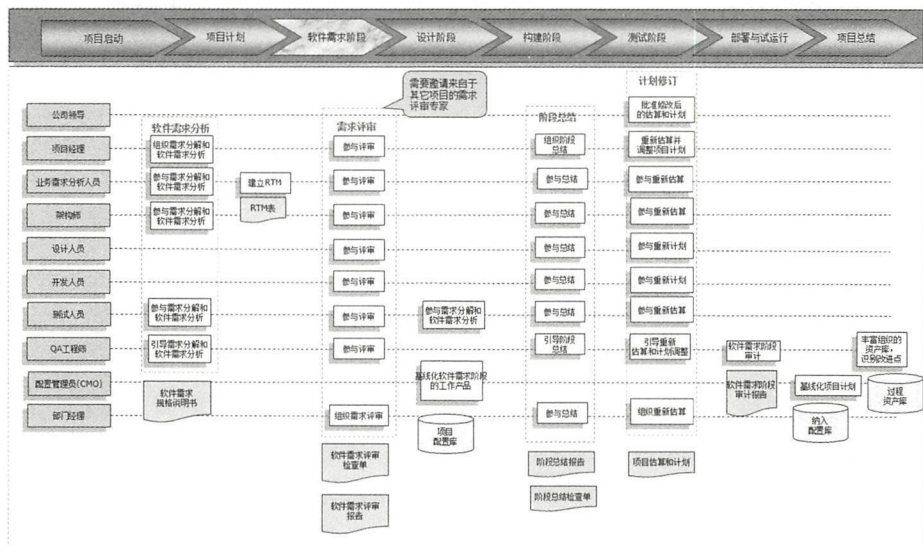


图 1-7 软件需求阶段



在一个项目中,任何人员,如项目经理、架构师、研发人员、测试人员、系统人员、DBA 等都可以外包。立项阶段、研发阶段、测试阶段和运维阶段等也都可以外包,唯一不能外包的就是需求。国外软件机构有过统计,项目失败 90% 以上的原因是需求问题。

需求分析阶段的产出是《软件需求规格说明书》和用户界面原型设计,这里着重讲解用户界面原型设计。如图 1-8 所示就是一个超市订货单的用户界面原型设计,读者可以进行参考。

订货单转送货单

订货单状态:未收货

ABC超市订货单

订货单编号: 2016010415424345

送货单编号: 3923576023862856

门店: 9999 上海陆家嘴店

下单日期: 2016.01.03

送货地址: 上海市陆家嘴路1号

部门: 10 饮料

厂商: 000001 上海第一有限公司

应到货日期: 2016.01.04

订货单差异回复截止时间: 2016.01.04 18:00

订单到达平台时间: 2016.01.04 15:42

订货取消日期: 2004.09.29

订货单未税总金额: 551.2923

订单备注: 自动订货生产送货单

订单差异回复已由门店商品循环部人员打印,不可进行回复!

订货单详细单据如下: 门店人员需要收货7箱,应付金额645.0093元

序号	条码	商品编号	子码	商品名称	型号	规格	包装	箱数	单位	零数	数量	单位	赠品	税率	未税单价	含税单价	未税金额	含税金额	备注	厂商商品编号
1	3920845600943	401097	001	蓝带蓝宝听啤		350ml	24	1	箱	0	24	罐	0	17%	2.3504	2.7800	56.4123	65.9997		
2	7931743880250	401192	001	百威啤酒听啤		500ml	24	1	箱	0	24	罐	0	17%	2.9915	3.5000	71.8045	84.0060		
3	5900669845194	800162	001	雪花清爽啤酒		1.45L	24	2	箱	0	12	瓶	0	17%	11.8385	13.5000	138.4600	161.9882		
4	2901382645397	201046	001	青岛冰爽啤酒		500ml	12	3	箱	0	18	瓶	0	17%	15.8120	18.5000	284.6221	333.0054		
共4种商品							总计	7		0	76			0			551.2959	645.0093		

请于送货前,确认订单之商品单价及税率与实际送货内容相符!于商品验收后之差异,不予以修改

1.不同订货单上的商品不要写在同一张送货单上
2.请务必于到货日期前送货
3.赠品请注明数量及品种

图 1-8 超市订货单用户界面原型设计

需求也是项目的灵魂,有了需求才有项目开展的可能。但初期的用户需求并不能作为项目实施的依据,最多可以作为项目的指导性意见或者方向,因为初期的需求绝大部分是由业务部门提出的,不够具体,功能点不明确,没有逻辑,也没有具体流程,在实际的项目执行过程中并没有多大的参考意义。所以,在有了初期的用户需求后,接下来就要进一步丰富和细化需求,每个功能点、计算逻辑、数据来源、处理流程、角色、权限、配置等,分析落地的可能性,然后进一步进行需求梳理、整合和丰富,进而加工成可以系统化的内容,进行产品选型,然后用技术实现。

基于以上内容,在实际的项目实施过程中需要注意以下几点。

(1) 需求的调研、挖掘和整理必须由项目经理牵头,由产品经理、业务需求方,甚至系统架构师一起完成需求的收集整理。当然,这里面要分甲方项目和乙方项目,基于笔者的经验,如果是甲方项目,建议需求完全由甲方的项目经理和产品经理负责跟业务需求方一起收集整理,收集整理完毕后可以交由供应商实施,也可交由自己的研发团队完成。如果是纯乙方项目,即甲方只有业务方,则乙方在收集整理甲方的需求时一定要注意需求范



围的控制，如果需求过多过大，建议分多期、多个阶段实施完成，一方面减少乙方的项目验收风险，另一方面缩短项目周期，也有利于保持项目团队的战斗激情。

(2) 功能点、数据计算逻辑、业务处理流程、权限控制和用户界面原型一定要跟业务部门相关需求方确认清楚，不管是甲方还是乙方的业务部门，让其对整理过的需求进行确认是很有必要的。

(3) 针对需求部分着重强调一点，在需求收集整理和挖掘的过程中，作为技术实施方的代表，尤其是产品经理将直接决定后期的项目实施团队是否存在项目延期的风险。所以，产品经理或者项目经理要负责把控用户的需求，尤其要引导客户的需求，整个项目实施过程就会相对轻松很多。

表 1-6 定义了需求阶段的条件、活动、标准输入和标准输出，并且确定了责任人。

表 1-6 需求阶段的活动和输入输出

入口条件	活动	出口条件	验证人
项目计划阶段结束	(1) 《软件需求规格说明书》通过评审； (2) 《软件需求规格说明书》是否基线化； (3) 用户界面原型图评审	(1) 《软件需求规格说明书》通过评审（附有“通过结论”的评审记录汇总表）； (2) 《软件需求规格说明书》有基线号； (3) 用户界面原型图评审通过（附有“通过结论”的评审记录汇总表或者会议纪要）	(1) 《软件需求规格说明书》：设计师、测试负责人。 (2) 用户界面原型图：开发经理、设计师、测试负责人、需求人员。 (3) 评审结论、是否基线化、会议记录、配置项是否填写：质量保证员。 (4) 配置项归档、基线化：配置管理员

1.4.4 变更管理

在整个软件开发过程中，需求变更会带来不确定性，但又不可避免。需求变更若无管理流程来引导和解决冲突，会导致开发测试得到的信息不完整，造成后续产品维护困难。

因此，既不能一概地拒绝需求变更要求，也不能一味地迁就变更要求，控制需求变更才是最好的应对策略。为了确保需求变更符合双方的利益，可以采取如下措施来控制需求变更。

- 分级管理需求变更

按照变更的影响程度和客户投入，可以分成关键性需求、后续关键性需求、后续重要需求、改良型需求和可选性需求。在时间优先级上进行管理和控制。



● 软件生命周期全过程需求变更管理

对一个需求分析做得很好的项目来说，需求规格说明书定义的范围越详细清晰，用户跟项目经理提出需求变更的几率就越小。

● 专人负责需求变更管理工作

使用需求管理工具来控制需求变更。设立专门的 CCB（变更控制委员会）对需求变更进行评审、裁定和验证。CCB 由项目所涉及的多方人员共同组成，包括用户方和开发方的决策人员，如项目经理、架构师、开发人员和测试工程师。

● 契约化管理需求变更

合作双方在签订协议之初，书面约定需求变更的提出方式、评价程序、修改要求、执行过程及验收要求等，只有这样才能确保需求变更按程序和要求有序进行。

● 需求变更信息化

需求变更必须提前沟通，双方要加强信息交换，防止搞突然袭击，更不能提出超越双方能力范围的需求变更。

如图 1-9 所示的变更管理流程运行图描述了各个角色在各个阶段的任务。

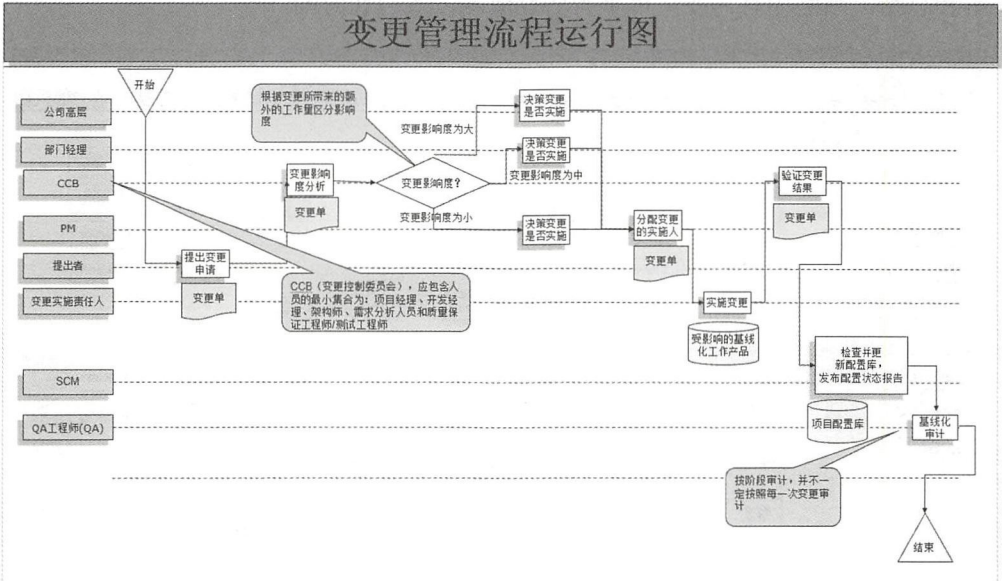


图 1-9 变更管理流程运行图

表 1-7 定义了变更阶段的条件、活动、标准输入和标准输出，并且确定了责任人。

表 1-7 变更阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 基线化文档有变动； (2) 项目范围有变动； (3) 项目进度变动； (4) 成本发生变动； (5) 配置项有变动； (6) 其他	(1) 发起变更流程； (2) 变更审批； (3) 实施变更； (4) 验证变更； (5) 归档或者是否基线化	输出《变更申请与处理表单》，并填写完整（含 CCB 签字认可）	(1) 《变更申请与处理表单》、基线化、配置项归档：配置管理员。 (2) 变更结论：质量保证员。 (3) 变更配置项验证：项目经理

1.4.5 设计阶段

软件设计的主要任务是把需求分析得到的结果转换为软件结构和数据结构，建立目标系统的逻辑模型，从而形成系统架构，明确软件系统应该“怎样做”。如图 1-10 所示列出了设计阶段各个角色的任务和产出。系统架构师和开发人员会做出一份《概要设计说明书》和《详细设计说明书》。

测试人员要根据需求文档细化系统测试、集成测试和单元测试的计划和用例设计，参与评审和总结。当然如有需要，可重新估算成本、人力和时间。

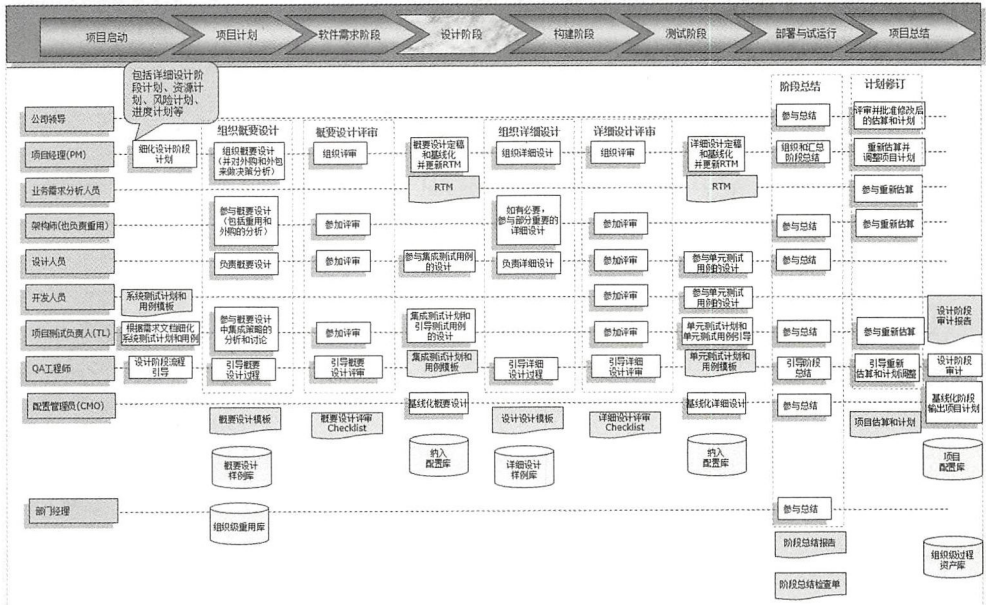


图 1-10 软件设计阶段

软件系统的设计是关系软件成败的重要因素。系统设计人员要专注于分析问题本身，挖掘重要的业务领域概念，并建立业务领域概念之间的关系，帮助用户及需求分析人员建立业务概念，确定用户业务的问题域和系统涉及的业务范围等，确定系统的整体架构，最终形成软件设计文档。

在设计阶段需要考虑如下因素：

- 从业务描述中提取名词；
- 从提取出来的名词中总结业务实体，区分名词中的属性、角色、实体、实例，形成问题域中操作实体的集合；
- 从业务实体集合中抽象业务模型，建立问题域的概念；
- 分析出模型实体，然后找出模型实体之间的关系；
- 用 UML 方法和图例进行领域模型设计，确定模型之间的关系；
- 在软件设计阶段需要保证各个模块的重用性和可扩展性；
- 运用设计模式封装变化，降低耦合，实现低耦合、高内聚；
- 通过面向对象思想解耦，将具体的东西抽象处理，分散的东西集中处理，解除对象之间的依赖。

表 1-8 定义了设计阶段的条件、活动、标准输入和标准输出，并且确定了责任人。

表 1-8 设计阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 项目软件需求阶段结束； (2) 《架构设计说明书》已经通过评审并基线； (3) 《软件需求规格说明书》已经通过评审并基线	(1) 《软件设计说明书》和《数据库设计说明书》评审； (2) 《软件设计说明书》和《数据库设计说明书》是否基线化； (3)集成测试用例和计划设计	(1) 《软件设计说明书》和《数据库设计说明书》通过评审（附有“通过结论”的评审记录汇总表）； (2) 《软件设计说明书》和《数据库设计说明书》有基线号； (3) 输出《集成测试计划和用例》	(1) 《软件设计说明书》、《数据库设计说明》、《集成测试计划和用例》：开发经理。 (2) 评审结论、是否基线化、配置项是否填写：质量保证员。 (3) 配置项归档、基线化：配置管理员

1.4.6 构建阶段

开发人员将上一个阶段详细设计的处理过程转换成计算机源代码，单元测试后提交给

测试人员执行必要的测试。测试人员需要协助开发人员对单元测试的计划和用例进行评审和指导，参与总结和重新估算。在构建阶段结束后，测试人员需要提供开发阶段的审计报告供项目经理做参考。如图 1-11 所示列出了构建阶段各个角色的任务和产出。

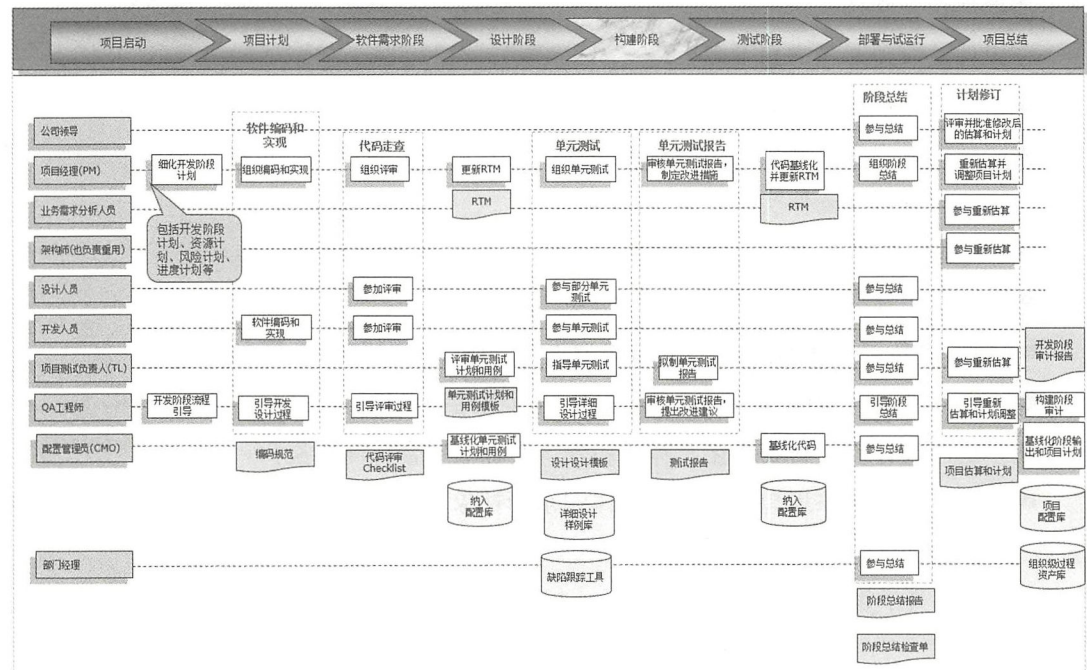


图 1-11 构建阶段

构建的规模大、项目多、速度快、业务复杂度高，针对这些问题，读者可以采用如下方法避免构建中的复杂性：

- 采用 MAVEN、ANT 等构建工具进行自动化构建，避免人为操作失误；
- 软件的包、版本进行统一管理，避免不同版本的直接冲突；
- 注意各个环境的一致性，保证构建脚本执行成功；
- 降低软件工程中模块构建的耦合度，提高软件功能模块构建系统的灵活性；
- 引用构建平台，监控构建日志，及时发现构建时的错误；
- 对构建失败进行总结分析，优化构建流程。

表 1-9 定义了构建阶段的条件、活动、标准输入和标准输出，并且确定了责任人。

表 1-9 构建阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 设计阶段结束； (2) 《软件设计说明书》已经通过评审并基线； (3) 《数据库设计说明书》已经通过评审并基线	(1) 代码实现； (2) 撰写用户手册； (3) 集成测试； (4) 交付给测试人员的代码是否基线化	(1) 输出《代码评审检查表》； (2) 输出《用户手册》（如果这个阶段未完成，最晚要在部署与试运行阶段前完成）； (3) 输出《集成测试报告》（附有“测试通过”的结论）； (4) 交付给测试人员的代码是否基线化	(1) 《代码评审检查表》、《用户手册》：开发经理。 (2) 《集成测试报告》：测试负责人。 (3) 是否基线化、配置项是否填写：质量保证员。 (4) 配置项归档、基线化：配置管理员

1.4.7 测试阶段

软件测试阶段的工作就是根据需求设计的测试方案和测试用例，利用人工或测试工具对产品进行功能和性能测试，需要跟踪故障缺陷，以确保开发的产品适合需求。如图 1-12 所示列出了测试阶段各个角色的任务和产出。测试人员在这个阶段需要准备集成测试和报告，之后准备系统测试计划和报告，并主持和参与系统测试，最后总结出系统测试评审报告。

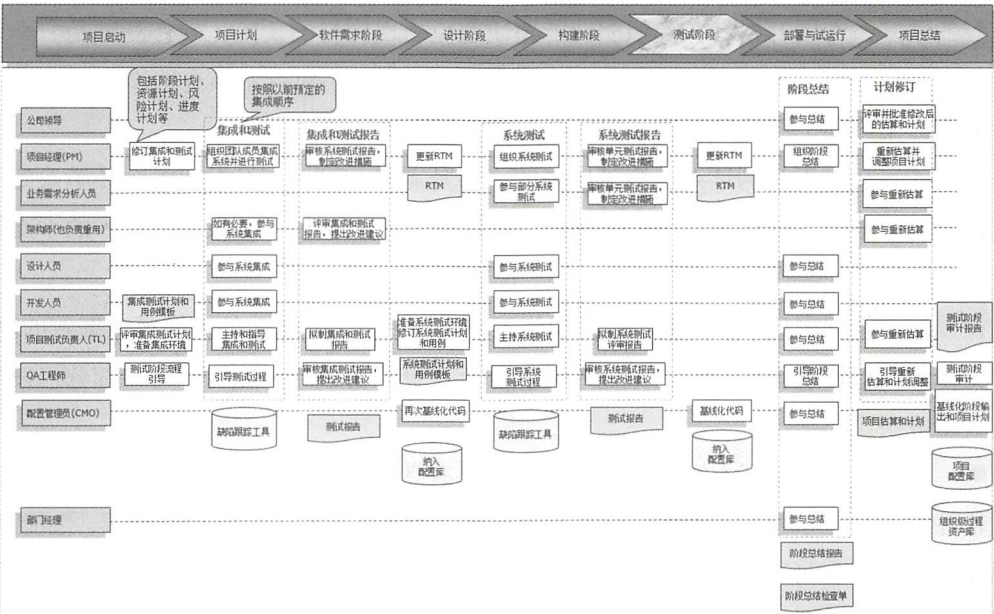


图 1-12 测试阶段

测试阶段需要测试人员来主导，开发人员配合修改缺陷，以确保产品质量。这里的测试主要包括代码扫描、功能测试、系统测试、集成测试、性能测试、安全测试和回归测试。

在软件测试过程中，经常会遇到如下一些误区。

- 开发人员测试自己编写的程序。

开发人员在编写程序时，有着固有的思维方式，也有可能理解错了需求定义和设计规范。若开发人员测试自己编写的代码，则很难发现问题。可能有些小公司没有配备专门的测试人员，但至少交叉测试也能找出比自我测试多得多的缺陷。

- 计划测试工作时假定不会发现错误。

这是非常危险的假设。人无完人，没有一个程序员可以 100%地保证产品没有缺陷。

- 测试在代码开发完后才开始。

测试是贯穿整个开发流程中的，比如，需求阶段要进行需求测试，拟定测试计划。越早进行测试就能越早发现缺陷，修正的成本也就越少。

- 测试就是为了找到缺陷。

这个错误观点强调了测试的目的在于要以查找错误为中心，而不仅仅是为了演示正确功能为目的。通过分析错误产生的原因和分布特征，可以帮助项目管理者发现当前软件管理过程的缺陷，帮助开发者在类似的功能模块避免出现同样的缺陷。软件测试是一个过程或一系列过程，用来确认软件完成其应该完成的功能并且不执行不该有的操作。

- 如果产品上线后发现质量问题，都是测试人员的错。

产品的高质量不是测试人员测试出来的，而是需求、设计、开发等各个环节决定的。出现错误不是某个人的错误，可能来源于混乱的项目管理，也可能来源于技术的不支持，还可能是环境的配置问题。

在发现质量问题后，要积极地分析和修正缺陷，对周边受影响的模块再做细致的测试，避免更多的类似错误仍然在线上。程序某部分存在更多错误的可能性，与该部分已发现错误的数量成正比。

- 开发人员对测试不够重视，觉得测试技术要求不高，随便谁都能做。

软件测试是一项极富创造性、极具智力挑战性的工作。测试包含功能测试、性能测试、自动化测试和安全测试等，这些都需要专业的技能，要设计覆盖率好的可重复执行的测试用例，还要不断更新新技术、新工具、新流程、新测试方法。

开发人员只有尊重测试人员，关系才能融洽，才有助于测试人员更加积极地测试产品，尽量发现更多的缺陷，确保产品质量。

● 开发延期，为了项目按时上线，压缩测试时间。

很多时候，由于需求变更过多或者开发技术限制等因素，代码交付的时候已经延期很多。而项目经理为了按时交付产品，会压缩整个测试时间，这样会使得测试人员的压力过大，为完成任务而跳过很多环节。在这种情况下，项目经理应该仔细评估风险和成本，可以延期项目，或者可以缩小第一期交付的产品特性，不牺牲产品质量。

测试人员在设计和执行测试时，需要秉承如下几个重要原则：

(1) 测试用例需清晰定义对预期的输入和输出；

(2) 应当彻底检查每个测试的执行结果；

(3) 测试用例的编写不仅应当根据有效和预料到的输入情况，而且也应当考虑无效和未预料到的输入情况；

(4) 检查程序是否“未做其应该做的”仅是测试的一半，测试的另一半是检查程序是否“做了其不应该做的”；

(5) 应避免测试用例用后即弃，除非软件本身就是一次性的软件。

如表 1-10 所示定义了测试阶段的条件、活动、标准输入和标准输出，并且确定了责任人。

表 1-10 测试阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 构建阶段结束； (2) 通过《集成测试报告》； (3) 《软件需求规格说明书》已经通过评审并基线	(1) 系统测试计划和用例评审； (2) 测试通过后的代码是否基线化	(1) 《系统测试计划》和《系统测试用例》通过评审（附有“通过结论”的评审记录汇总表）； (2) 输出《系统测试报告》（附有“测试通过”的结论）； (3) 对测试通过后的代码是否基线化	(1) 《系统测试计划》、《系统测试用例》：测试负责人 (2) 《系统测试报告》：项目经理。 (3) 评审结论、是否基线化、配置项是否填写：质量保证员。 (4) 配置项归档、基线化：配置管理员

1.4.8 部署与试运行

部署主要由系统运维人员搭建部署环境，提供给测试人员回归测试、验收系统。项目

经理需要配合部署人员做项目部署，了解项目部署环境，跟踪项目运行期间产生的缺陷，安排相关人员对相应缺陷进行更改。如图 1-13 所示列出了部署与试运行阶段各个角色的任务和产出。

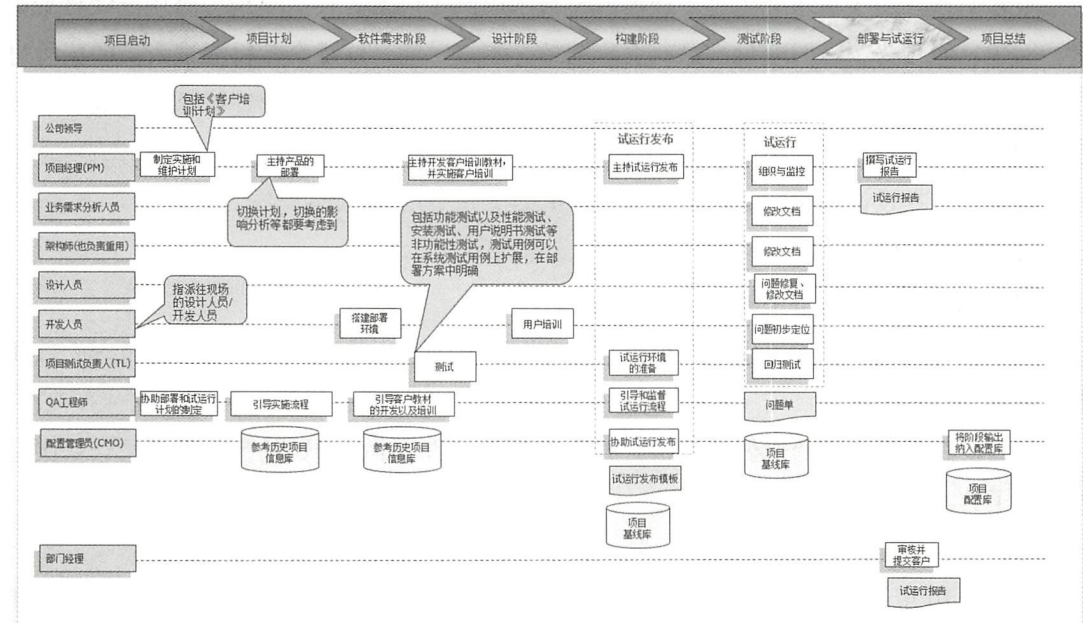


图 1-13 部署与试运行阶段

在部署与试运行阶段，有以下几个常见的问题。

- 应急响应系统不完善，没有自动化警报。

有些公司没有设立自动化运维系统，在系统异常时，没有设定警报。在早期阶段，运维人员需要值班，以保证试运行系统出故障可以马上处理。但这不是长久之计，还是需要自动化警告，通过 QQ、邮件、微信等方式通知运维人员，提高响应速度。

- 消极应对异常信息，态度不端正。

运维人员如不能对异常信息进行有效处理，难以分清轻重缓急，就会延误解决时间。这里需要运维负责人对组内人员做好培训，强调优先级。另外，项目经理需要快速跟进这些问题，必要时告知运维负责人分配和处理这些问题。

表 1-11 定义了部署与试运行阶段的条件、活动、标准输入和标准输出，并且确定了责任人。

表 1-11 部署与试运行阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 测试阶段结束； (2) 有通过的《系统测试报告》； (3) 用户手册	(1) 部署方案； (2) 试运行； (3) 得到客户确认	(1) 输出《部署方案》； (2) 《用户验收报告》； (3) 《用户确认书》	(1) 《部署方案》：开发经理。 (2) 《用户验收报告》、《用户确认书》：项目经理。 (3) 配置项是否填写：质量保证员。 (4) 配置项归档、基线化：配置管理员

1.4.9 项目总结

项目在试运行之后，最好由第三方进行验收测试，并根据验收报告进行整改。验收成功后，正常上线运行。各部门召开会议，进行总结回顾，列出优点、不足和报告总结，确定哪些可以在二期中进行提高和完善，以及如何更好地加强部门之间的协作等。

在项目总结阶段，可以得到如下总结：

- 《验收测试报告》；
- 《项目总结报告》；
- 《项目发布报告》；
- 《项目结项审计报告》；
- 归档检查单；
- 项目总结检查单；
- 总结经验教训，丰富资产库。

项目总结阶段的流程如图 1-14 所示。

项目总结会应避免出现以下几种情况。

- 总结会过场子。
- 大家讲些不痛不痒的话，没有真正总结优缺点。
- 总结会变成批斗会。

由于项目延期上线或者试运行问题较多，大家互相数落对方的不是，这样不利于团队合作。项目开发过程中暴露出来的问题，需要就事论事，不要进行人身攻击。应该把之前的经验教训都记录下来，避免其他项目再走同样的弯路。

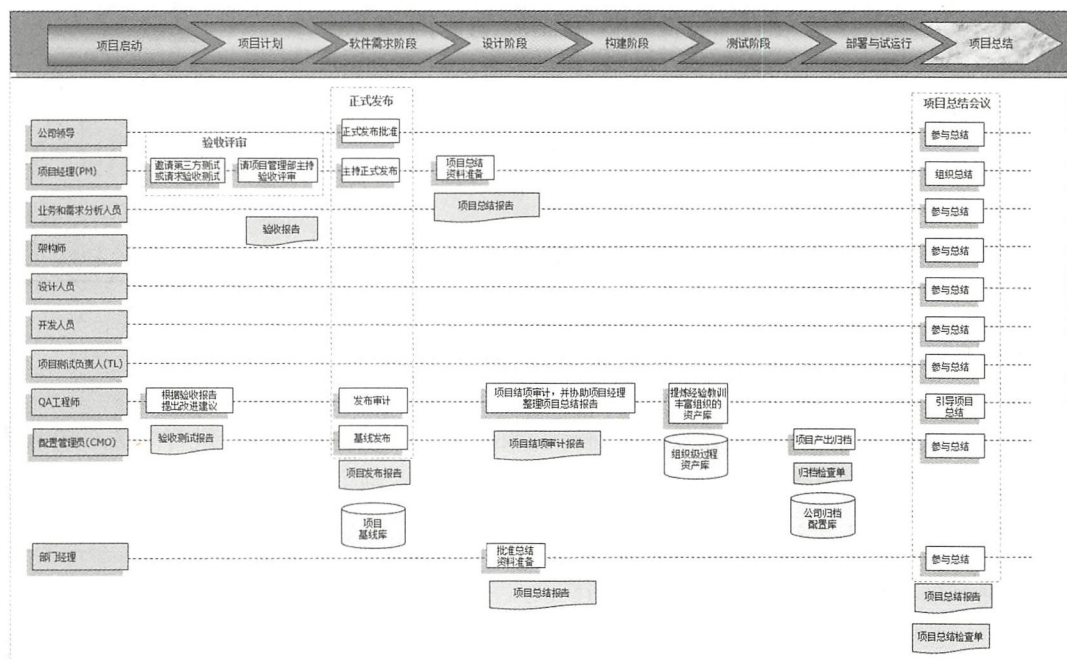


图 1-14 项目总结阶段

表 1-12 定义了项目总结阶段的条件、活动、标准输入和标准输出,并且确定了责任人。

表 1-12 项目总结阶段的活动和输入输出

入口条件	活动	出口条件	验证人
(1) 部署与试运行阶段完成； (2) 由客户签字验收	(1) 申请项目总结； (2) 项目总结会议	(1) 《项目发布报告》； (2) 所有缺陷都关闭； (3) 《里程碑成本管理手册》内容完整； (4) 风险全部关闭	(1) 《项目发布报告》、《里程碑成本管理手册》、缺陷关闭、风险关闭：项目经理。 (2) 配置项是否填写：质量保证员。 (3) 配置项归档、基线化：配置管理员

1.5 项目管理十诫

笔者参与过大大小小几十个项目，包括 B2B 沃尔玛供应链、SAP 各个模块实施和 Abap 二次开发、第三方支付平台、清结算和风险控制系统，总结出一些经验。由于自身的局限性，以及项目的不同、技术的不同和参与人员的不同，与读者要做的项目可能有很大的差

异，以下这些经验仅供参考。

(1) 绝对不要出现如图 1-15 所示的情况，图中所有人将全部工作都推给开发人员，这样会造成项目内各成员关系紧张，不利于沟通和后续工作的进行。其实很多工作是在其他阶段进行的，例如公共类和框架标准都是在设计阶段完成的。



图 1-15 项目管理常见问题

(2) 很多项目都是在需求未定的情况下就开始进行研发工作，这是最大的问题。相当多的项目经理在开发部门工作完成后，发现与需求不一致，再重新研发，宁愿花费几个月的时间反复研究，也不愿意在前期把需求定义分析清楚。最不能接受的情况是研发测试完成后再修改需求，本末倒置。

(3) 当项目立项或者启动后，读者会发现戴明环、PMP 等各种软件方法论和软件管理方法都无法施展，因为没有有一个方法论或者软件开发过程是适合自己公司、自己团队的，只能在不断的磨合中创新一套自己公司、自己团队的软件开发过程。总之，适合自己的方法论就是最好的方法论，成功实施的软件开发过程就是最好的软件开发过程。

(4) 在团队会议中，类似“这么简单的项目都做不了”的话说了也没有用，权利是吓不住项目组成员的，大家期望项目经理以身作则，而不是作威作福。公司的规章制度一定要严格遵守，项目经理应该起模范作用。

(5) 不要轻易承诺，比如“我做过多个优秀项目，这个不成问题”“这个项目是老板定

的，大家放心，资源肯定能落实到位”。盲目乐观的结果注定是悲观，需要落实的奖励机制和奖金政策一定要落实。

(6) 有时常听到类似“兄弟，我的前程就靠你了”“今年的年终奖就看这个项目了，一起吃个饭，把这个事情搞定”这样的话。孤注一掷的成功概率往往低于 50%，因为风险都在一个人的身上，这恰恰是最高风险。

(7) 没有计划，工作估算就是随便拍脑袋想想，例如“你说 10 个工作日，7 个工作日吧，没有问题，你们是最棒的，能力最强，加几天班，肯定能完成”。在项目启动阶段，计划、策略、风险控制和预防机制才是最关键的。在估算各自部门的时间时，尽量有一个时间缓冲，要不然会造成措手不及。

(8) 遇到错误，不要说“你辜负了我的信任”。错误已经发生，要做的是找到解决方案，埋怨只会破坏和谐，团队氛围越来越差。

(9) “都是项目经理的问题，大不了我走人，绝对不和你一个项目组了。”三十六计“走为上策”不一定是正确的，下一个项目、下一个公司、下一个项目经理可能更差。在项目进行中，参与的项目组成员应该是固定的，尽量不要被临时抽调，否则会造成延迟。

(10) 软件配置管理一定要严格，发布和确认一定要有说明。

1.6 项目管理工具对比

项目管理软件在软件开发、团队管理中是必不可少的工具，它可以帮助项目经理更有效地执行项目计划，支持项目管理、资源计划、文档、Wiki、协作、问题跟踪等。下面是几款较流行的项目管理工具，分别简要介绍一下。

1.6.1 Microsoft Project

Microsoft Project 是微软公司出品的一款项目管理软件，可以利用日历和甘特图来展示项目进度。使用它还能做出漂亮的图表，查看工时、资源和成本等报告。其界面如图 1-16 所示。

开源协议：商业软件。

官方网址：<https://products.office.com/zh-cn/Project/>。

Microsoft Project 具有以下几个主要特性。

- 轻松规划和管理项目，帮助取得预期业务价值和控制成本。

- 提高效率, 明确优先顺序, 界面展示日常工作、项目任务、重要详细信息和多个日程表。
- 简化资源管理, 管理日常工作或临时项目的工作状态。
- 几乎可以在任意位置进行管理, 通过与 Office 或 SharePoint 之间的集成分享最新状态、对话和 Project 时间表。
- 实时沟通, 可以与同一个办公室或全球任意角落的团队成员实时沟通。

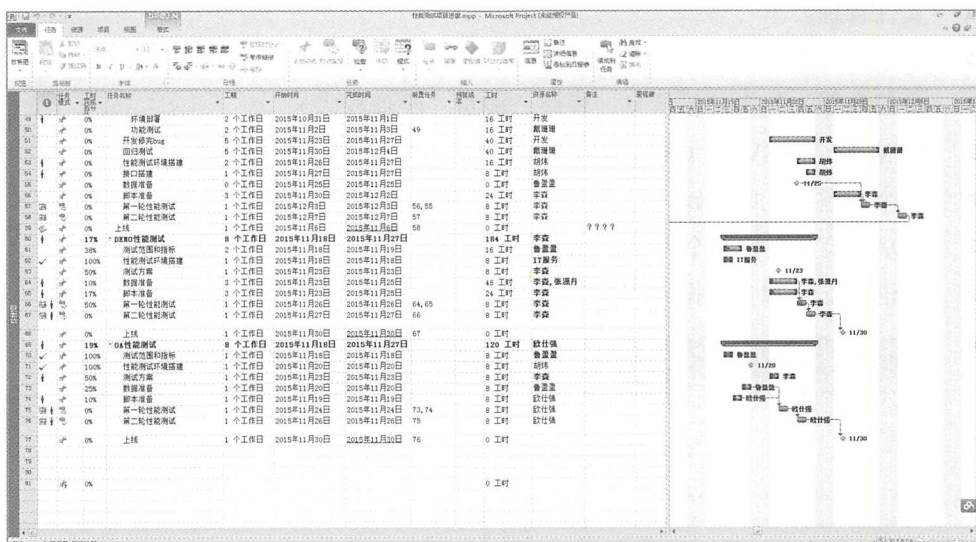


图 1-16 Microsoft Project 界面

1.6.2 Redmine

使用 Ruby on Rails 框架开发的 Web 项目管理软件, 支持多种数据库, 跨平台, 可以提供 Wiki、新闻、文档和文件管理, 可以集成版本管理系统 (如 Git、SVN) 等。本系统也会利用日历和甘特图来展示项目进度。其界面如图 1-17 所示。

开源协议: GPL。

在线演示: <http://demo.redmine.org>。

官方网址: <http://www.redmine.org/projects/redmine/wiki/Download>。

Redmine 有以下几个主要特性。

- 支持多项目。

- 活动管理。
- 角色基于权限控制管理。
- 问题跟踪管理。
- 可利用甘特图和日历。
- 新闻、文档管理。
- 每个项目具有 Wiki 和讨论区。
- 文件管理。
- 版本管理和问题类别（普通权限）。
- 项目配置，包含项目信息、模块、成员（经理权限）。
- SCM 整合（SVN、CVS、Git 等）。
- 单个问题的工时跟踪。
- LDAP 验证支持和个人注册机制。
- 多语言支持和多数据库支持。

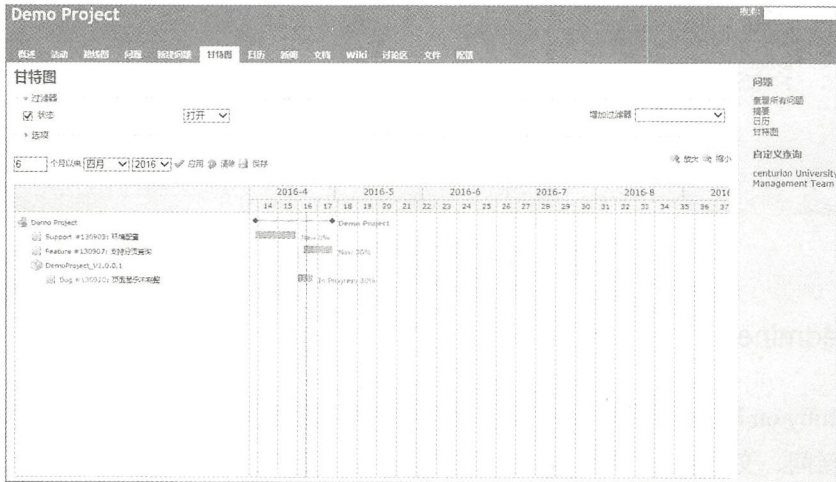


图 1-17 Redmine 界面

1.6.3 Feng Office

Feng Office 是一款网上办公系统，它将项目、任务管理、部门管理、内容管理、文档、甘特图和日历等多个实用功能集成在一起，非常适合中小企业的团队协作及项目管理。目

前，NASA、NBA 等组织正在使用该系统。

开源协议：AGPL。

在线演示：<http://www.fengoffice.com/web/demo.php>。

该系统分为社区版和专业版，社区版和专业版的特性区别如图 1-18 所示。

Main features	Community Edition	Professional Edition	Enterprise Edition
Workspace Management	✓	✓	✓
Project Management	✗	✓	✓
Client Management	✗	✓	✓
Documents	✓	✓	✓
Notes	✓	✓	✓
Calendar	✓	✓	✓
Timesheet	✓	✓	✓
Automatic alerts and reminders	✓	✓	✓
Contact management	✓	✓	✓
Tags	✓	✓	✓
Basic reports	✓	✓	✓
Task templates	✓	✓	✓
Workflow processes	✓	✓	✓
Basic e-mail	✗	✓	✓
Group Mailing	✗	✓	✓
Mail Rules	✗	✓	✓
Task dependencies	✗	✓	✓
Advanced reports	✗	✓	✓
Gantt Chart	✗	✓	✓
Mobile Module (HTML5)	✗	✓	✓
Expenses Module	✗	✗	✓
Objectives Module	✗	✗	✓

图 1-18 Feng Office 不同版本的特性区别

如图 1-19 所示是 Feng Office 界面，可以看到项目管理、日历管理、活动和用户等。

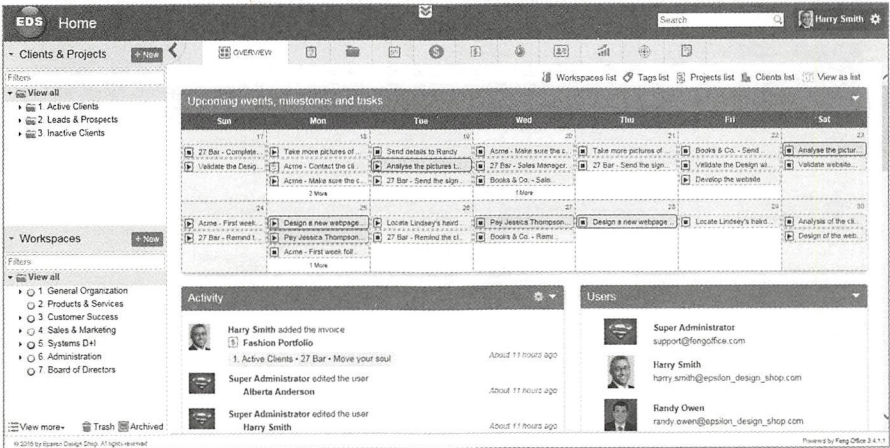


图 1-19 Feng Office 界面

1.6.4 ProjQtOr

ProjQtOr 的名字来源于 “Quality based Project Organizer”，是一个开源的基于项目管理的质量控制软件，适合软件项目，也可以用于其他类型项目。

开源协议：GPL。

在线演示：<http://demo.projqt.org/view/main.php>。

官方下载网址：<http://www.projqt.org/en/product-en/downloads>。

ProjQtOr 主要有以下几个特点。

- 安装简单。

ProjQtOr 使用 PHP、MySQL 和 Ajax。

- 可参数化。

每个用户参数和值都可通过前端页面显示，方便进行更改。

- 可定制化。

由于 ProjQtOr 是开源的框架型的软件，因此可以根据自己的需要扩展或修改内容。

- 方便监控。

可以进行连接数管理，查看在线会话数，也可以关闭会话，可以在维护阶段启动或关闭应用。

- 使用方便。

ProjQtOr 界面友好，支持多语言，也支持多种浏览器，如 IE、Firefox、Chrome。还可根据个人喜好，选择不同的主题风格，其界面如图 1-20 所示。

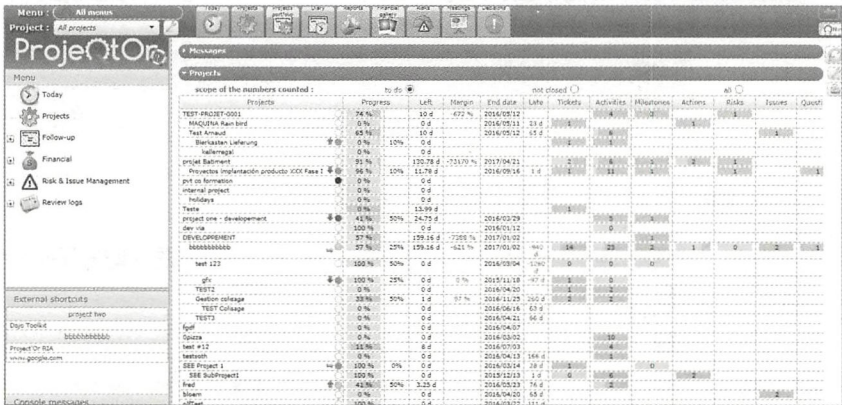


图 1-20 ProjQtOr 界面

1.6.5 Project-Open

Project-Open 是一个开源的、基于 Web 的项目管理应用程序，涵盖多方面，如客户关系管理、销售、项目、财政管理和任务管理。

开源协议：GPL。

官方示例网站：<http://demo.project-open.net/>。

官方下载网址：<http://www.project-open.com/en/list-installers>。

如图 1-21 所示是 Project-Open 界面，可以看到项目管理、CRM 和用户管理等。

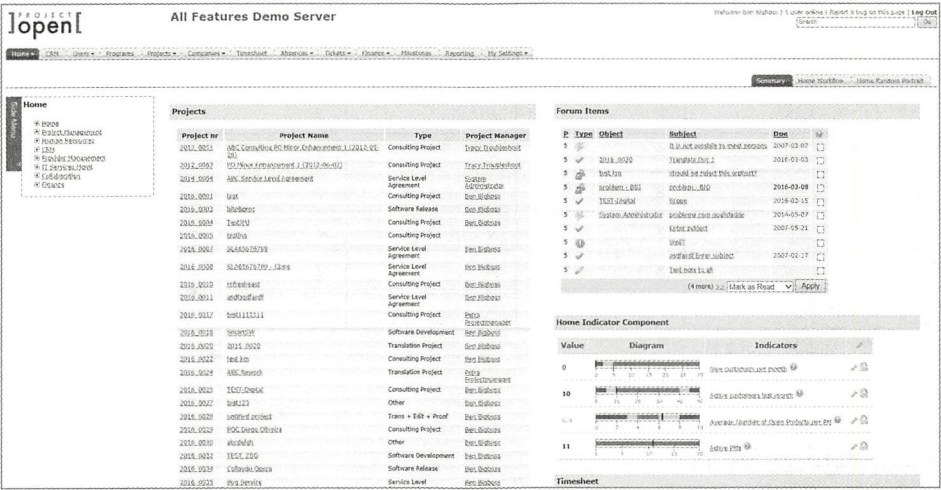


图 1-21 Project-Open 界面

1.6.6 各管理工具特性对比

对以上 5 个工具的特性进行对比，如表 1-13 所示。综合考虑，本书选用了 ProjeQtOr 这款开源的软件，下一节会详细介绍这款软件。

表 1-13 项目管理工具对比

项目管理软件名称	MS Project	Redmine	ProjeQtOr	Feng Office	Project-Open
开源/商业	商业版	开源	开源	社区版免费，企业版收费	开源
架构	.Net	Ruby	PHP+MySQL	PHP+MySQL	PHP+MySQL
项目管理	√	√	√	√	√

续表

项目管理软件名称	MS Project	Redmine	ProjeQtOr	Feng Office	Project-Open
活动管理	※	√	√	√	√
任务管理	※	※	√	√	√
甘特图	√	√	√	√	√
日历	×	√	√	√	√
需求管理	×	×	√	√	√
流程管理	×	×	√	√	√
工时管理	×	※	※	√	√
文档管理	×	※	√	√	√
资源管理	√	×	√	√	√
联系人管理	×	×	√	√	√
客户管理	×	×	√	√	√
风险管理	×	※	√	√	√
财务成本管理	×	×	√	√	√
报表	√	×	√	√	√
移动端模块	√	×	×	√	×
目标管理	√	×	×	√	×

注：×不支持 ※部分支持 √支持

1.7 ProjeQtOr 简介

很多项目经理只关注了项目计划，而忽略了其他方面，造成项目延期，出现较大风险。一个优秀的项目经理需要预估项目风险、评估风险，建立执行计划和迁移计划，跟踪记录项目中发生的情况，如事件（故障）、缺陷、变更申请、支持申请等，而 ProjeQtOr 恰恰是提供这些功能的一个平台。下面简要介绍部署和使用 ProjeQtOr。

1.7.1 部署 ProjeQtOr

（1）部署 ProjeQtOr 之前需要搭建以下环境：

- MySQL 数据库 5 及以上版本，或者 PostgreSQL 数据库 V8.4、V9.1 以上；
- PHP 服务器 5.2 及以上版本；

- 一个 WebServer，如 Apache、Nginx。

提示：需要快速搭建环境，可以使用一键式的 XAMPP（Apache+MySQL+PHP+PERL）建站集成软件包、Easy PHP、Zend Server。这里就不具体讲解安装步骤了，读者可自行从网上查看。这种方式仅适合测试评估使用，实际上线环境还是建议安装独立的 HTTP、PHP 服务器和数据库，以达到较好的性能。

(2) 从 ProjQtOr 官网下载需要安装的版本，当前比较稳定的版本是 V5.2.5。

下载的网址是：<http://www.projektor.org/en/product-en/downloads>。

(3) 将下载的文件（projektorVx.y.z.zip）解压到需要安装的 Web 服务器的 Web 目录下。若使用 XAMPP 安装，其解压目录类似于“/opt/lampp/htdocs/projektor”。赋予 ProjQtOr 目录足够的权限，可以满足日志文件的创建与写入。

(4) 在确认服务器和数据库启动的状态下，可以访问浏览器 <http://127.0.0.1/projektor>。这里的“127.0.0.1”可以改为计算机的真实 IP 地址。

(5) 第一次访问 ProjQtOr 时，会弹出如图 1-22 所示的初始配置页面，可以进行各项配置。

如果需要重新设置配置项，可删除 projektor/tool/parametersLocation.php 文件。升级系统时，在对配置页面进行设置后，数据库会升级，原有数据仍会保留。

图 1-22 ProjQtOr 初始配置页面 1

如图 1-23 所示的初始配置页面中的配置项可根据需要进行修改，如 E-mail、Default time zone（时区）和 Currency（货币）。根据需要可以设置语言为“zh”，表示简体中文；时区



设置成“Asia/Shanghai”，表示上海；货币设置成“¥”，表示人民币。

eMail address of sender : a valid email as sender for mailing function

eMail address to reply to : a valid email to define the reply to for mailing function

eMail of administrator : a valid email of the administrator (will appear on error messages)

SMTP Server : localhost address of SMTP (mail) server, may be left blank (default is 'localhost')

SMTP Port : 25 port to talk to SMTP (mail) server (default is '25')

Sendmail program path : to set only on issue to send mails, or not using default sendmail

Default password for initialization : projector any string possible as default password

Min length for password : 5 any integer, to force a long password (keep is reasonable)

Default locale to be used on i18n : en default language, 'zh' for Chinese - 简体中文, 'hr' for Croatian - Hrvatski, 'nl' for Dutch - French - Français, 'de' for German - Deutsch, 'el' for Greek - Ελληνικό, 'it' for Italian - Italiano, 'ru' for Russian - Русский, 'es' for Spanish - Español, 'ua' for Ukrainian - Українська

Default time zone : Europe/Paris default time zone, list can be found at <http://us3.php.net/manual/en/timezones.php>

Currency : € currency displayed for costs

Currency position : after position of currency displayed for costs

Use fading mode for frames refresh : true 'true' or 'false', if set to 'true' screens will appear in a fading motion

Icon size on menu tree : 22 '16' for small icons, '22' for medium icons, '32' for big icons

Default color theme, proposed while login : ProjQtOr select a theme in the list

Directory to store Attachments : ../files/attach/ any valid directory, set to empty string to disable attachment

Max file size for attachment : 2097152 **Security hint :** move it outside web access

Temp directory for reports : ../files/report/ any valid directory in the web structure

Memory limit for PDF reports : 512 any numeric value, for size in MB

Log file name : ../files/logs/projector_\$(date).log any valid file name, may contain '\$(date)' to get 1 file a day

Log level : 2 **Security hint :** move it outside web access

Parameter file name : ../files/config/parameters.php a php file name where to store parameters

Security hint : move it outside web access

图 1-23 ProjQtOr 初始配置页面 2

配置页面上的所有配置项都存储在 parameters.php 文件中，对应关系如表 1-14 所示。这里只列举了部分参数配置项，其他配置项可查看用户手册。

表 1-14 ProjQtOr 配置项

配置项	描述	参数名
Database type	数据库类型，支持 MySQL 或者 PostgreSQL	\$paramDbType
Database host	MySQL 或 PostgreSQL 服务器名（默认为 localhost）	\$paramDbHost
Database port	MySQL 或 PostgreSQL 服务器端口号	\$paramDbPort
Database user to connect	数据库用户（默认为 root）	\$paramDbUser
Database password for user	数据库用户的密码	\$paramDbPassword
Database schema name	数据库实例名（默认为 projector）；如果没有数据库实例，会新建一个实例	\$paramDbName
Name to be displayed	识别数据库连接，任意值皆可，显示在登录后的页面底部	\$paramDbDisplayName
Database prefix for table names	表名前缀，可以为空	\$paramDbPrefix





续表

配置项	描述	参数名
Default locale to be used on i18n	设置国际化默认语言, 但用户参数页面可以覆盖此设置。常用的值有: en (默认值, 英语)、zh (中文)、fr (法语)	\$paramDefaultLocale
Default time zone	默认时区(默认值为 Europe/Paris), 中国时区可设置为 Asia/Shanghai	\$paramDefaultTimezone
Currency	默认货币	\$currency

(6) 设置配置项后, 进入登录页面, 输入默认的管理员用户名 admin 和密码 admin, 单击“OK”按钮。这时将会花费几分钟的时间来创建数据库, 完成后会提示更新成功信息, 如图 1-24 所示。

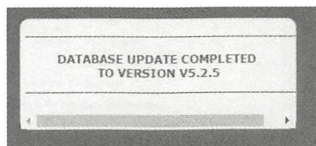


图 1-24 数据库更新成功

(7) 登录系统, ProjeQtOr 初始页面如图 1-25 所示。

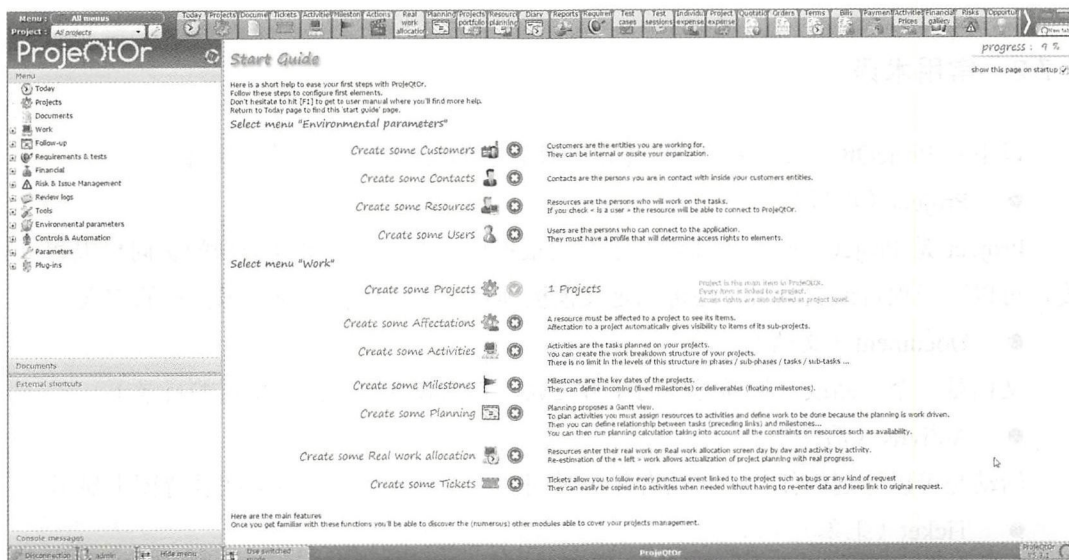


图 1-25 ProjeQtOr 初始页面

(8) 登录系统后, 可以更改系统语言。这里选择左侧菜单项中的“Parameters”→“User





parameters”，在右侧打开的用户参数配置项中，将“language”设置为“Chinese-简体中文”，如图 1-26 所示。若还有部分选项没有翻译成中文，可以自行修改/tool/i18n/nls/zh/lang.js 文件。

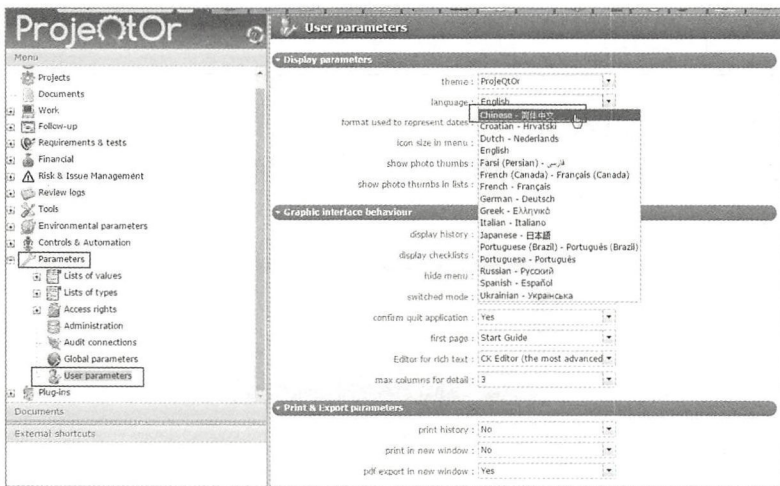


图 1-26 用户参数—语言设置

1.7.2 常用术语

以下是 ProjQtOr 系统中常见的术语，理解它们有助于更好地管理项目。

- Project（项目）

Project 是 ProjQtOr 最主要的元素。它根据用户概况提供最高级别的访问权限和可见度，可以有子项目或父项目。还可以定义模板项目，不过仅是公司组织结构的定义。

- Document（文档）

文档是一个产品或项目描述性的参考文献，它以版本号存储在特定的目录下。

- Activity（活动）

活动是可以计划的任务，一般有起止时间，可以分配资源，也会在甘特图上显示。

- Ticket（工单）

工单是一种不能统一规划的任务。一般来说，每个工单都是短期的活动，需要反馈和跟踪。比如，不能预估的缺陷也可以定义成一个工单，因为其不能预计到，也需要反馈和记录。一个权限申请也可以作为一个工单。





- Milestone（里程碑）

里程碑是一个项目中关键性的、可以作为标志的事件。它在项目中不占用资源，是一个时间点。

- Actions（措施）

措施用来规避风险，给未发生的风险提供迁移措施。

- Risk（风险）

风险会对项目造成负面影响和威胁，但可以被中和或者降低。风险管理是项目管理中非常重要的一点，可以识别影响，避免和监控一些风险。

- Opportunities（机遇）

机遇可以看做是正面的风险，它不是危险而是机会，可以给项目提供正面影响。



- Issue（问题）

问题是项目中存在的一个风险。

1.7.3 系统配置—系统管理员

前面讲述了安装和配置 ProjeQtOr，下面讲解使用管理员账号创建基础数据的方法，为后续的项目管理使用做准备。

1. 创建角色

用系统管理员账号登录系统后，选择菜单项“参数（Parameters）”→“访问权限（Access Rights）”→“产品（Profiles）”，进入用户角色信息页面，单击“添加”按钮，如图 1-27 所示。填写正确的角色描述信息后，单击“保存”按钮或按 Ctrl+S 快捷键添加角色成功。

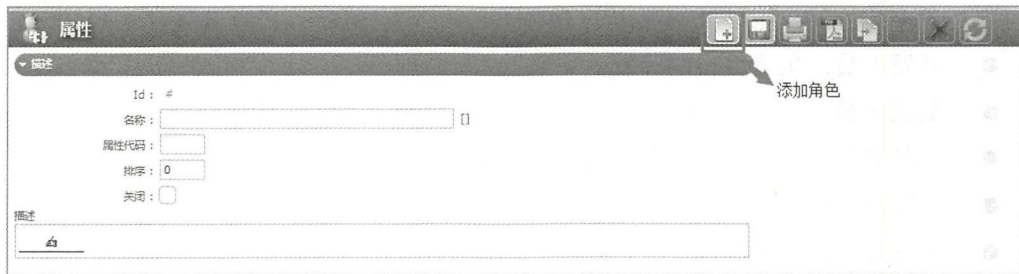


图 1-27 添加角色





提示：ProjeQtOr 的所有条目保存都可以用 Ctrl+S 快捷键。

系统默认已有的角色是项目成员（Project Member）、项目经理（Project Leader）和总监（Supervisor），用户可根据项目需求添加开发人员、测试人员、开发经理、测试经理、产品专员和产品经理等角色，如图 1-28 所示。



Id	名称
15	[SCM]
14	[SQA]
13	[产品经理]
12	[产品专员]
11	[测试经理]
10	[开发经理]
9	[测试人员]
8	[开发人员]
1	系统管理员
2	总监
3	项目经理
4	项目成员
6	外部项目经理
7	外部项目成员
5	项目来宾

图 1-28 添加角色

2. 给角色分配权限

用系统管理员账号登录系统，选择菜单项“参数（Parameters）”→“访问权限（Access Rights）”→“访问表格（Access to form）”，进入访问权限设置页面，对前面创建的角色赋予权限。各个角色的权限分配如图 1-29 所示。

- 开发人员：负责项目的开发。
- 测试人员：负责项目的测试。
- 产品专员、产品经理：负责提出产品需求与验收。
- 开发经理：分配开发任务。
- 测试经理：分配测试任务。
- 项目经理：跟踪项目的进度，协调项目组成员之间的工作。
- 软件质量保证员：负责项目质量管理和里程碑跟进。





- 软件配置管理员：负责配置管理。

访问表格													
	产品经理	开发人员	开发工程师	测试人员	测试工程师	系统管理员	总编	项目经理	项目成员	外部项目经理	外部项目成员	项目库员	项目专员
今日数据	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
读库	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
文档	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
工作													
工单	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Tickets (sample)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
活动	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
跟踪单	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
任务	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
报告													
实际工作	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
规划	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Projects portfolio	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
资源计划	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Diary	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
报告	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
项目和模式													
需求	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
测试用例	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
测试用例	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

图 1-29 各个角色的权限分配

选择菜单项“参数 (Parameters)”→“访问权限 (Access Rights)”→“登录模式 (Access Modes)”，可以查看登录模式匹配的不同阅读权、创建权、更新权、删除权和次序，如图 1-30 所示。

登录模式						
Id	名称	阅读权	创建权	更新权	删除权	次序
1	许可	自身项目的要素	无要素	无要素	无要素	100
2	Reader+	全部项目全部要素	无要素	无要素	无要素	150
3	更新	自身项目的要素	无要素	自身项目的要素	无要素	200
4	Updater+	全部项目全部要素	无要素	全部项目全部要素	无要素	250
5	创建	自身项目的要素	自身项目的要素	自身要素	自身要素	300
6	Creator+	全部项目全部要素	全部项目全部要素	自身要素	自身要素	350
7	管理	自身项目的要素	自身项目的要素	自身项目的要素	自身项目的要素	400
8	Manager+	全部项目全部要素	全部项目全部要素	全部项目全部要素	全部项目全部要素	450
10	Reader own	自身要素	无要素	无要素	无要素	900
9	禁止登录	无要素	无要素	无要素	无要素	999

图 1-30 登录模式页面

选择菜单项“参数 (Parameters)”→“访问权限 (Access Rights)”→“数据访问模式 (Access to data)”，进入数据访问模式页面，可以为各类角色分配对应的数据访问权限，如图 1-31 所示。






质量全面管控：从项目管理到容灾测试



图 1-31 数据访问模式页面

3. 创建用户

用系统管理员账号登录系统，选择菜单项“环境参数”→“用户”，进入新建用户页面，如图 1-32 所示。单击  按钮添加用户，填写用户名，设置属性和其他描述信息。然后单击“重置密码”按钮，重置密码后，单击“OK”按钮，再单击“保存”按钮即可。新增用户的默认密码是“projektor”，用户第一次登录时会被强制要求修改密码。

添加新用户时可以添加用户为联系人或资源，勾选相应的单选框即可。图 1-32 中的联系人和资源可以分配给项目、活动和工单。

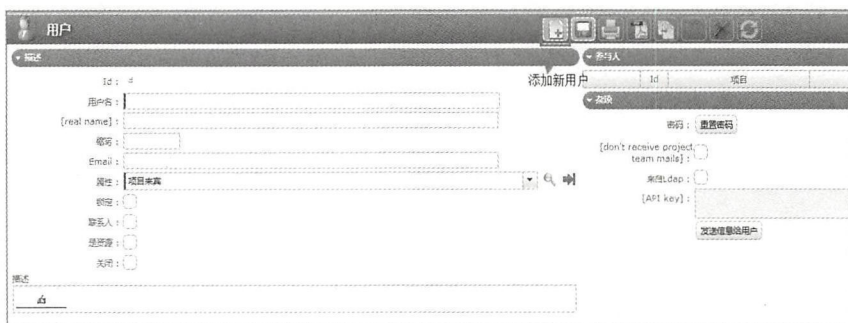
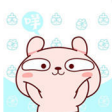


图 1-32 新建用户页面

新建的用户会显示在用户列表中，如图 1-33 所示。





Id	用户名	[photo]	属性
11	张开理	[icon]	[开发经理]
10	袁总监	[icon]	总监
9	徐产理	[icon]	[产品经理]
8	鲁测试	[icon]	[测试人员]
7	李开发	[icon]	[开发人员]
6	葛测试	[icon]	[测试经理]
5	张产品	[icon]	[产品专员]
4	luyingying150925	[icon]	[测试人员] 鲁晶晶
3	郝项理	[icon]	项目经理
2	guest	[icon]	项目来宾

图 1-33 用户列表

如表 1-15 所示列出了用户、资源和联系人的各种可能组合。



表 1-15 URC 矩阵

	连接 ProjeQtOr 应用	数据可见	资源可用
URC	√	√	√
UR	√	√	√
UC	√	√	×
U	√	√	×
R	×	×	√

注：U = User（用户），R = Resource（资源），C = Contact（联系人）

1.7.4 需求管理—产品经理

1. 新增产品

用产品经理账号登录系统，选择菜单项“环境参数（Environment Parameters）”→“产品（Products）”，其页面如图 1-34 所示。单击按钮，依次输入名称、客户、发包方、附件和描述等信息，单击“保存”按钮新增产品。

- 客户（Customer）：来源于“环境参数”→“客户”的数据。
- 发包方（Prime contractor）：来源于“环境参数”→“联系人”的数据。
- 版本（Version）：来源于“环境参数”→“版本”的数据。






质量全面管控：从项目管理到容灾测试



图 1-34 产品页面

2. 新增版本

产品经理创建完产品后，选择菜单项“环境参数”→“版本（Product Versions）”，打开新增版本页面，如图 1-35 所示。单击  按钮，产品需关联已创建的产品，输入名称、发包方、责任人等相关信息，单击“保存”按钮，创建版本成功。创建成功后的版本号可以在原有的产品中看到，即在产品页面的“产品版本”区域中可以看到。

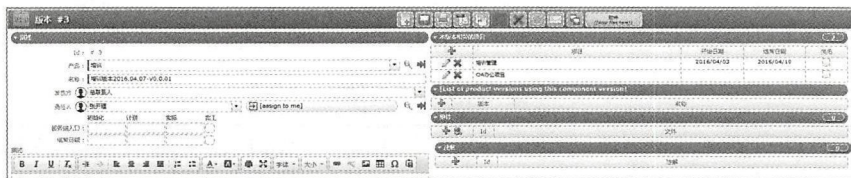


图 1-35 新增版本页面

3. 新增需求


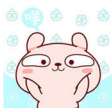
选择菜单项“需求和测试（Requirements & test）”→“需求（Requirments）”，显示需求详情页面，如图 1-36 所示。单击  按钮，输入名称、关联产品或项目、需求类型、描述和责任人等相关信息，单击“保存”按钮。




图 1-36 需求详情页面



提示：如果项目已经新增，请在此关联项目。如果项目未新增，请在新增项目的同时关联需求。

1.7.5 项目管理—项目经理

1. 新增项目

用项目经理账号登录系统后，选择菜单项“项目（Projects）”，单击“新建项目”按钮，打开如图 1-37 所示的项目详情页面。输入名称、类别、客户、开始日期、结束日期和成本等相关信息后，单击“保存”按钮即可。

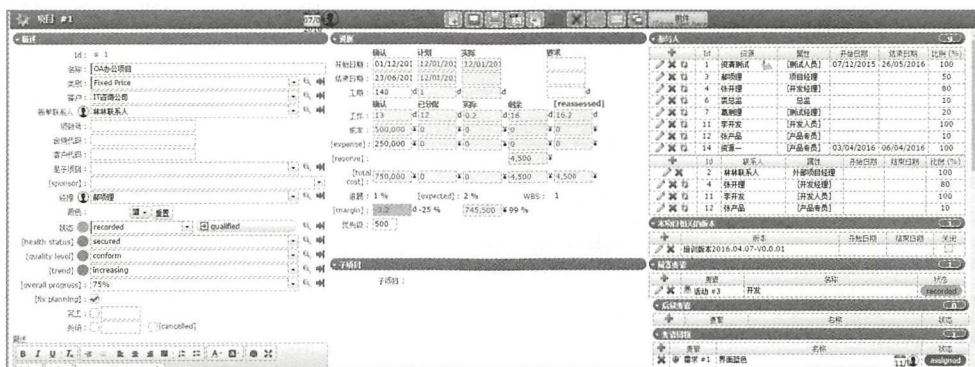


图 1-37 项目详情页面

如表 1-16 所示列出了项目的一些元素。

表 1-16 项目的元素


项目元素	描述
Requested start date （请求开始日期）	期望开始日期
Requested end date （请求结束日期）	期望结束日期
Requested duration （请求持续时间）	期望持续时间（按工作日）
Validated start date （已验证开始日期）	项目不能迟于此日期开始
Validated end date （已验证结束日期）	项目不能迟于此日期结束
Validated duration （已验证持续时间）	项目时间不能多于此期限
Validated work （已验证工作）	项目验证的总工作时间
Assigned work （已分配的工作）	项目分配的总工作时间，只读
Planned start date （计划开始日期）	计划的开始日期，只读

质量全面管控：从项目管理到容灾测试

续表

项目元素	描述
Planned end date（计划结束日期）	计划的结束日期，只读
Planned duration（计划持续时间）	计划的持续时间，只读。 “Planned duration” = “Planned end date” - “Planned start date”（按工作日，可按任何单位来计算）
Planned work（已计划的工作时间）	计划的总工作时间，只读。 “Planned” = “Real” + “Left”
Real start date（实际开始日期）	在“实际工作分配”页面中输入的第一个实际任务开始的日期。只读
Real end date（实际结束日期）	如果项目状态是“done”（结束），那么这是在“实际工作分配”页面中输入的最后一个实际任务结束的日期。只读
Real duration（实际持续时间）	实际持续时间“Real duration” = “Real end date” - “Real start date”（按工作日）。只读
Real work（实际的工作）	实际工作量。只读
Left work（剩余的工作）	项目中剩余的工作。只读
Work Breakdown Structure（工作分解结构）	以可交付成果为导向对项目要素进行分组，归纳和定义项目的整个工作范围，每下降一层代表对项目更详细的定义。WBS 以数字标识层次
Progress（实际进度）	项目的实际进度，以百分比计算，实际进度% = “实际工作时间” / “已计划的工作时间” × 100%
Expected（期望进度）	项目的期望进度，以百分比计算，期望进度% = “实际工作时间” / “已计划的工作时间” × 100%

2. 关联资源（参与人）

选择对应项目，单击“参与人新增”按钮，在打开的对话框中输入资源、属性、比例、开始日期、结束日期和描述等信息，如图 1-38 所示，然后单击“OK”按钮保存。

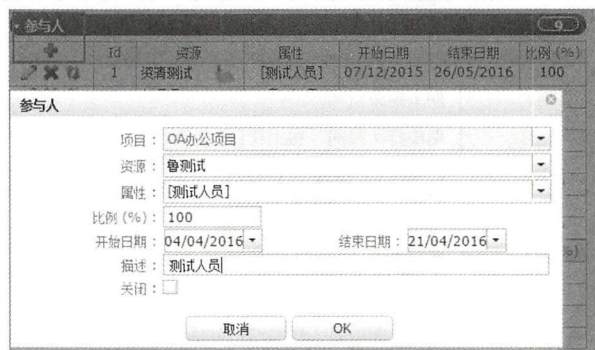


图 1-38 “参与人”对话框

3. 关联版本


选择菜单项“项目”，再选择对应的项目，在“本项目相关的版本”区域中，单击“新增”按钮，在打开的对话框中关联对应的版本，输入开始日期与结束日期，如图 1-39 所示，单击“OK”按钮保存。



图 1-39 “项目—版本链接”对话框

4. 关联需求

选择菜单项“项目”，再选择对应的项目，在“要素链接”区域中，单击“新增”按钮，在打开的对话框中将“链接要素类型”设置为“需求”，如图 1-40 所示，单击“OK”按钮保存。这里的“链接要素类型”除了“需求”外，还可以选择其他类型，如项目、活动、工单等。

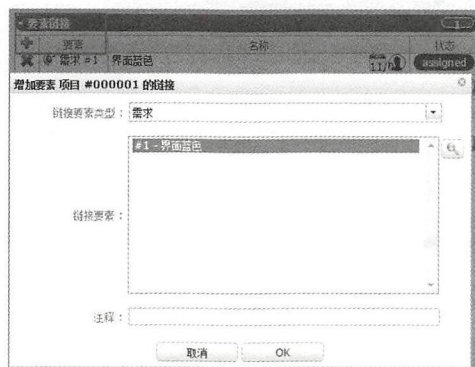


图 1-40 关联需求

5. 新增工单

用项目经理账号登录系统后，选择菜单项“工作”→“工单”，再单击“新增”按钮，

质量全面管控：从项目管理到容灾测试

在打开的页面中输入工单名称、序列类型、项目、紧急、请求者等相关参数，如图 1-41 所示。还可以在此页面中修改工单状态。

图 1-41 新增工单页面

6. 工单关联需求

用项目经理账号登录系统后，选择菜单项“工作”→“工单”，选择对应的工单要素链接，单击“增加”按钮，在打开的对话框中关联需求，如图 1-42 所示，单击“OK”按钮即可。

图 1-42 工单关联需求

1.7.6 开发管理—开发人员

1. 查看工作台

用开发人员账号登录系统，在主界面中选择“今日概要”，可以查看自己负责的项目，

还可以指派任务、设置任务提醒等，如图 1-43 所示。

The screenshot shows the Projector web application interface. It includes a header with the Projector logo and a welcome message. Below the header, there are several sections: a task list, a task details section, and a task history section. The task list shows tasks with columns for ID, Name, Status, and Progress. The task details section shows the details for a specific task, including its name, status, and progress. The task history section shows a list of tasks that have been completed.

ID	名称	状态	进度
12	任务管理	Task	100%
13	任务管理	Task	100%
14	任务管理	Task	100%
15	任务管理	Task	100%

图 1-43 查看工作台

2. 新增活动

单击菜单项“工作”→“活动”，再单击“新增”按钮，在打开的页面中输入名称、活动类型和项目等相关信息，如图 1-44 所示，单击“保存”按钮。

The screenshot shows the 'Add Activity' page in the Projector web application. The page has a header with the title '活动 #22' and a sub-header '0001-TAS-1'. The main content area is divided into two sections: '活动信息' (Activity Information) and '活动状态' (Activity Status). The '活动信息' section contains fields for '名称' (Name), '活动类型' (Activity Type), '项目' (Project), '外部参照' (External Reference), '请求者' (Requester), and '来源' (Source). The '活动状态' section contains fields for '父活动' (Parent Activity), '状态' (Status), '责任人' (Responsible Person), '办理' (Processing), '完工' (Completion), '关闭' (Close), and '目标版本' (Target Version). The '状态' field is set to 'recorded'. The '责任人' field has a button '[assign to me]'. The '办理' field has a button '[cancelled]'. The '目标版本' field has a dropdown menu.

图 1-44 新增活动页面

3. 指派责任人

选择一个活动，再选择该活动对应的责任人，接着输入开始日期、结束日期、成本等相关信息，如图 1-45 所示，然后单击“保存”按钮。

4. 任务状态修改

选择菜单项“工作”→“活动”，在打开的页面中根据项目完成的情况，填写实际项目

质量全面管控：从项目管理到容灾测试

的开始日期、结束日期和状态等信息，如图 1-46 所示。

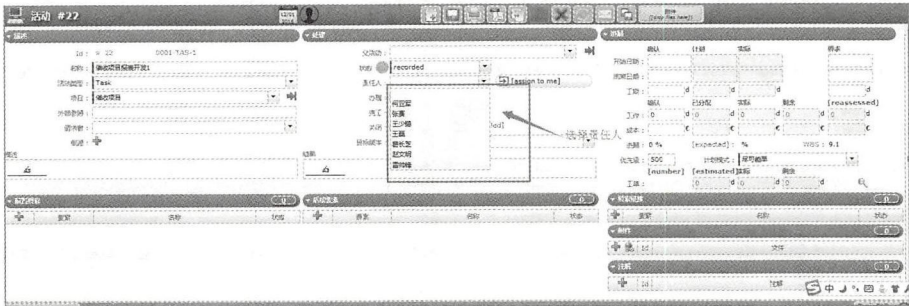


图 1-45 指派责任人页面



图 1-46 任务状态修改页面

1.7.7 测试管理—测试人员

1. 查看工作台

用测试人员账号登录系统，选择今日概要，可以查看自己负责的项目、指派任务和设置任务提醒，如图 1-47 所示。

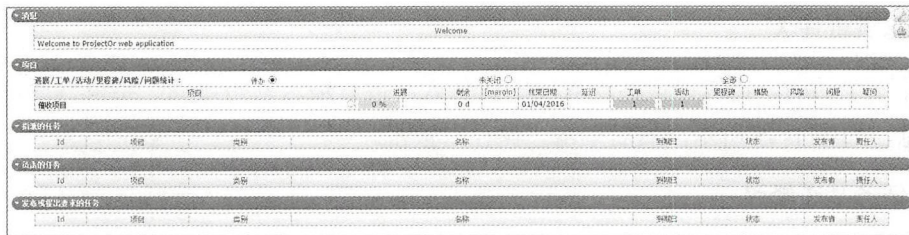


图 1-47 测试人员工作台

2. 新增活动

选择菜单项“工作”→“活动”，单击“新增”按钮，在打开的页面中输入名称、活动类型和项目等相关信息，如图 1-48 所示，单击“保存”按钮。



图 1-48 测试人员新增活动页面

3. 指派责任人

选择一个活动，在打开的页面中选择该活动对应的责任人，接着输入开始日期、结束日期、成本等相关信息，然后单击“保存”按钮。



图 1-49 指派责任人页面

4. 任务状态修改

选择菜单项“工作”→“活动”，打开如图 1-50 所示的页面，根据项目的完成情况输

入实际项目的开始日期、结束日期、状态等信息，单击“保存”按钮即可。



图 1-50 任务状态修改页面

1.8 要点回顾

- 项目管理的定义、发展史、管理的模型和认证方式。
- 质量管理和项目管理是不可分割的。
- 项目管理流程经历项目启动、计划阶段、需求管理、变更管理、设计阶段、构建开发、测试阶段、部署与运行和项目总结几个阶段。了解各个阶段中每个角色的责任和产出。
- 常用的几个项目管理工具的对比。
- 了解 ProjeQtOr 中的常用元素，明白 Project、Activity、Ticket、Document、Milestone、Action、Risk、Opportunity 和 Issue 代表什么。
- 配置管理 ProjeQtOr。

第 2 章

项目需求管理

Frederick Brooks（弗雷德里克·布鲁克斯）在 1987 年经典文章 “No Silver Bullet（没有银弹）” 中阐述了需求的重要性：“开发软件系统最困难的部分就是准确说明开发什么。最困难的概念性工作是编写出详细的需求，包括所有面向用户、面向机器和其他软件系统的接口。此工作一旦做错，将会给系统带来极大的损害，并且以后对它的修改也极为困难，软件项目中 60% 的问题都是在需求阶段埋下的祸根。”

美国纽约有一个“失败产品博物馆”，展出的产品有 80 000 多件，有大公司生产的，也有小公司生产的，有的功能十分强大，有的超前意识特别强。这些产品失败的原因各种各样，但最主要的就是产品功能脱离消费者需求。

需求的重要性不言而喻，是一个项目的核心。软件工程任何阶段，如设计、研发、测试、运维、部署等都可以进行外包，唯一不能外包的就是需求，这是项目的根本。

本章将介绍如下内容：

- 需求的定义和需求类别；
- 需求开发和需求管理；
- 收集需求，编写需求说明书；
- 测试需求；
- 用 Plandora 进行需求管理。

2.1 认识需求

软件需求是软件项目成功的关键，实践一再证明，这是一个关键的阶段，读者要从根

本上理解需求。下面重点讲解需求阶段的一些关键点。

2.1.1 需求的基本定义

软件行业刚刚起步时，需求的定义是这样描述的：“用户所需要的并能触发一个程序或系统开发工作的说明”。

需求分析专家 Alan Davis 在 1993 年扩展了这个概念：“从系统外部能发现系统所具有的满足于用户的特点、功能及属性等”。这个定义强调的是产品是什么样的，而并非产品是怎样设计、构造的。

1997 年，IEEE 软件工程标准词汇表中第一次记录了需求的基本定义。

软件需求是：

- (1) 用户解决问题或达到目标所需的条件或能力；
- (2) 系统或系统部件要满足合同、标准、规范或其他规定文档所需的条件或能力；
- (3) 一种反映上面 (1) 或 (2) 所描述的条件或能力的文档说明。

广泛地讲，需求来源于用户的一些“需要”，这些“需要”被分析、确认后形成完整的文档，该文档详细地说明了产品“必须或应当”做什么。

关键的问题是一定要编写需求文档。如果只有一堆邮件、贴条、几次会谈或一些零碎的对话，就确信自己已明白用户的需求，那完全是不切实际的。

2.1.2 需求类别

软件需求说明书的编制是为了使用户和软件开发者对该软件的初始规定有一个共同的理解，使之成为整个开发工作的基础。软件需求说明书包含硬件设备、功能特性、性能指标、输入输出、接口需求、上下级系统、警示信息、保密安全、数据与数据库、文档和法规的要求。软件需求分为以下三类，如图 2-1 所示。

- (1) 业务需求：包括业务需求规范、项目视图等。

首先由市场人员捕获最原始的需求，这是用客户的语言、客户的业务和流程来做的。再由需求分析人员将客户的需求进行整理，反映出客户对系统、产品高层次的目标要求，通常在项目定义与范围文档中予以说明。

- (2) 用户需求：包括用户需求说明、实例文档。描述用户使用产品必须要实现的需求，

应在业务流程图和解决方案中详细说明。

(3) 软件需求：包括功能需求和非功能需求，以需求规格说明书的形式记录需求。

- 功能需求：定义开发人员必须实现的软件功能，使用户利用系统能够完成他们的任务，从而满足业务需求。系统需求是功能需求的一个子集，包括模型、解决方案、可行性分析报告和风险估计。
- 非功能需求：是指软件产品为满足用户业务需求而必须具有除功能需求以外的特性，包括系统的性能、安全性、可靠性、可维护性、可扩展性等。

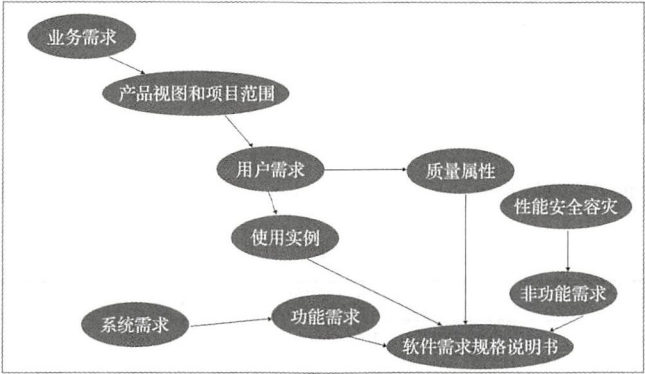


图 2-1 软件需求类型

性能测试需求是非功能需求中的重要子集，具体的性能需求指标可参考表 2-1。

表 2-1 性能需求指标

需求项	实例
系统是否会有交易峰值的出现	例如未来接入大商户、做市场活动等
是否需要支持多用户并发访问	例如在某个时刻用户集中使用
是否对系统处理能力/时间有明确要求	业务上对交易处理时间/笔数有明确要求
是否有大数据量的任务需要处理	例如批量付款之类的业务，且提交处理的数据量较大
是否有长期需后台执行的任务	例如数据库 Job、Quartz 需长期后台执行
是否涉及大数据量的交易表操作（读写）	例如对关键数据表有读写操作
是否为核心交易系统	例如人民币网关、交易引擎，占公司营收比重较大的部分
是否采用分布式部署且数据通信量较大	主要考虑横向扩展带来的开销和是否存在网络瓶颈
是否为基础应用/服务平台	为其他系统提供服务，其处理能力和稳定性影响其他系统
是否与多个外部系统有调用	考虑多个系统之间的网络通信及数据库资源争用
是否对系统可用性、连续性有较高要求	视具体业务需求而定

续表

需求项	实例
是否有异常状态下的应对措施/补偿机制	在一定压力下的异常处理机制
是否对系统稳定性有较高要求	需要保证长时间运行无故障，针对故障比较敏感的系统

提示：在将用户需求向软件需求转换的过程中，通常采用“使用实例”的方法获取软件的需求。“使用实例”通过描述“系统”和“活动者”之间的交互来描述系统的行为。通过分解这些行为，可以梳理出执行这些需求的所有步骤。它最主要的优点是用户导向的，用户可以根据自己所对应的“使用实例”来不断细化自己的需求。此外，“使用实例”还可以方便地得到系统功能的测试用例。

2.1.3 需求干系人

由于需求的重要性，所以相关干系人比较多，具体的角色和承担的职责如表 2-2 所示。

表 2-2 需求干系人

角色	职责
部门经理	(1) 参与《用户需求说明书》和《需求规格说明书》的评审。 (2) 对需求开发和管理中的一些重大活动进行监督
客户代表	(1) 配合进行需求调研，提供所需调研资料。 (2) 提出、确认系统需求。 (3) 参与需求评审
项目经理	(1) 制定需求管理计划。 (2) 分配资源，进行需求开发活动。 (3) 确保整个项目生命周期中需求管理工作的实施
需求人员	(1) 进行需求分析。 (2) 编写《用户需求说明书》和《需求规格说明书》。 (3) 组织对需求文档进行评审。 (4) 根据需求基线创建《需求跟踪矩阵》。 (5) 维护《需求跟踪矩阵》，保证项目分析设计成果与需求一一对应
开发人员	参与《用户需求说明书》和《需求规格说明书》的评审
测试人员	(1) 参与《用户需求说明书》和《需求规格说明书》的评审。 (2) 分析《需求规格说明书》，编写《测试计划》
配置管理员	(1) 建立需求基线。 (2) 对需求开发和管理过程中的所有文档进行配置管理
质量保证员	客观评价需求开发与管理的活动和产品

2.2 需求工程

所有与需求直接相关的活动统称为需求工程。需求工程分为需求开发和需求管理，如图 2-2 所示。其中，需求开发包括需求调查、需求分析和需求定义，需求管理包括需求确认、需求跟踪和需求变更控制。开发出来的需求是整个项目开发活动和管理活动的需求。描述清晰的需求活动贯穿整个项目开发周期，进一步支持产品开发和管理的。

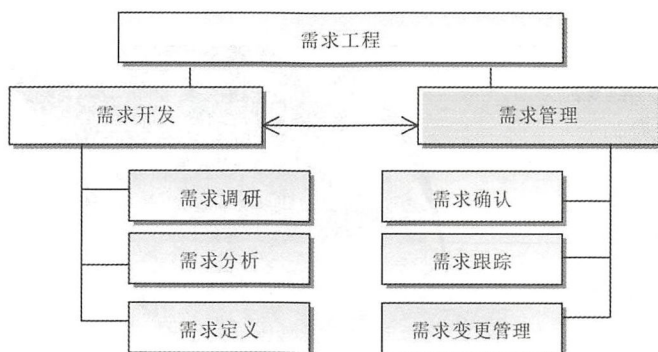


图 2-2 需求工程

需求人员可以把需求作为一个项目，获取需求的途径是从用户那里进行调研。互联网企业是产品经理根据市场需求和行业动态进行设计，自己提出需求，召开需求说明会议，讲解《需求规格说明书》和 UI 界面原型，参与人员包括项目经理、产品经理、架构师、数据库管理员、开发人员、测试人员、运维人员、系统人员甚至是客服人员。

团队成员根据需求人员的讲解基本了解需求后，各自根据角色的不同进行消化和吸收，提出问题并记录下来。

需求人员会召开需求评审会议，至少 3 轮甚至更多，反复地分析和修改，最后进行确认。

当多方达成共识后，配置管理人员定义基线纳入版本库，最后首席技术官或者高层批准启动项目。

2.3 需求开发

整个需求开发过程需要通过调研和分析获取用户需求，生成《用户需求规格说明书》，

并且把用户需求转换为产品需求，如图 2-3 所示。

- 需求调研的目的是通过各种途径获取用户的需求信息（原始材料），生成《用户需求说明书》。
- 需求分析的目的是对各种需求信息进行分析，消除错误，描述细节等。
- 需求开发的目的是通过调查和分析，获取用户需求并定义产品需求。

经过以上 3 个阶段，就可以非常顺利地转换为软件产品需求，生成《软件需求规格说明书》，以便开展设计研发和测试工作。

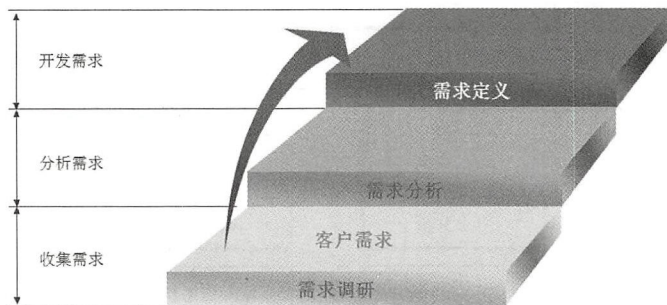


图 2-3 需求开发过程

2.3.1 需求调研

用户讲不清楚需求是普遍现象，这是让需求调研人员头痛的大问题。有些用户真的不知道需求是什么，或者对需求只有朦胧的感觉，讲不清楚需求。

例如，前些年全国各地很多政府机关推行自动化办公建设。这些机关的领导和办公人员大多数已经习惯了纸质办公，还不清楚自动化办公的作用，就请软件公司的需求人员替他们总结需求。在这种情况下，需求人员绝不能以用户讲不清楚需求为借口而草率地对待需求开发工作，否则会导致完成的项目根本无法满足政府机关的需求，后果是极其严重的。

无论是什么原因导致用户讲不清楚需求，需求分析员必须设法搞清楚用户真正的需求，这是需求分析人员的职责，也是职业的挑战。

一般情况下，大多数公司的项目可以分成两类：新产品项目和升级类项目。

- 新产品项目：产品经理根据《产品市场需求文档》，在允许的时间、成本和性能要求下，分析每项需求实施的可行性，明确与每项需求实现相关的风险，包括与其他需求的冲突、对外界因素的依赖和技术障碍。新产品项目的需求可能来源于市

场部，也可能来源于高层，还可能来源于具体用户。

- 升级类项目：根据项目所涉及的产品线，选用相应的《产品实施模板》和《用户环境调研模板》，同用户或需求发起方沟通项目需求，编写《业务需求说明书》。经项目经理确认是升级类项目后，产品经理需根据《业务需求说明书》编写《软件需求规格说明书》。升级类项目的需求可能来自市场人员的分析、关键客户的反馈等。当然，也可能来源于项目组内部，比如开发人员、测试人员和客服人员等。

接下来重点讲解一下新产品项目，因为开发新的项目要拜访客户，要进行现场调研。

首先清楚用户在需求工程中的“权利”。

- 有权要求开发方派遣有资质的需求分析人员和相关人员。
- 有权要求开发方采用用户熟悉的语言来描述需求，即开发方必须提供用户看得懂的需求文档。
- 有权审查需求文档，并对有争议的需求做出决策。如果认为需求文档不能准确地反映用户真实的意愿，可以拒绝在需求文档上签字。
- 如果用户想要变更需求，有权要求开发方对该变更产生的影响做出真实可信的评估，以便用户决定是否实施变更。

提示：用户是需求的提出者，所有的研发和测试工作都是为了满足客户的需求，所以用户是“上帝”。同样，用户也是有义务的，用户在需求工程中的“义务”就是“与需求人员共同评审需求文档，确保需求文档准确地反映用户真实的意愿，并且一定要签字确认，这个最重要”。

为了让读者更清楚在需求调研阶段如何开展工作，笔者梳理出如表 2-3 所示的需求调研过程中的输入输出和步骤，供读者参考。

表 2-3 需求调研中的输入输出和过程

输入项	前期准备与项目相关的材料
主要步骤	<p>第一步：编写需求调研计划。</p> <p>第二步：调研与记录。</p> <p>第三步：详细记录用户需求的信息。</p> <p>第四步：撰写《用户需求说明书》。</p> <p>第五步：进行需求评审</p>
输出项	《用户需求说明书》
结束原则	需求人员编写《用户需求说明书》，项目经理带领团队进行第一次需求评审，并且配置管理工程师把《用户需求说明书》纳入配置库

需求调研的关键点如下：

- 一定要提供给客户原型，由客户确认项目的整体风格；根据行业特点，原型要借鉴目前行业流行的设计方法，还有想法超前的界面设计方式。可以设计几套原型，让客户进行选择，调研的时候尽量带上资深的前端工程师，这样做的好处是可以现场改正。需求调研要直到客户满意为止，因为前端界面的不确定性对后期的研发和测试会有很大影响。
- 针对不同的用户进行讲解和培训，收集结果，不断地进行现场改进。操作流程一定要简化，使用户不经过系统的培训就可以按照界面的设计完成操作。
- 不要聘请行业专家，因为行业专家是少数，他们的技能是非常熟练的。而公司研发的项目对应的对象是要求操作简捷的最终用户，双方的关注点是不同的。
- 需求人员在调研前期，要认真学习客户的业务，这将有助于需求人员设计出真正满足用户需求的产品。产品中还要使用客户的行业术语，例如应收账款、应付账款。

如果不了解客户的业务，在没有任何准备的情况下进行需求调研，那么结果肯定会造成如图 2-4 所示的问题。客户描述的需求被需求人员曲解，如果再被开发人员理解成另一个需求，那么最后的产品离客户的真正需求将相差甚远。

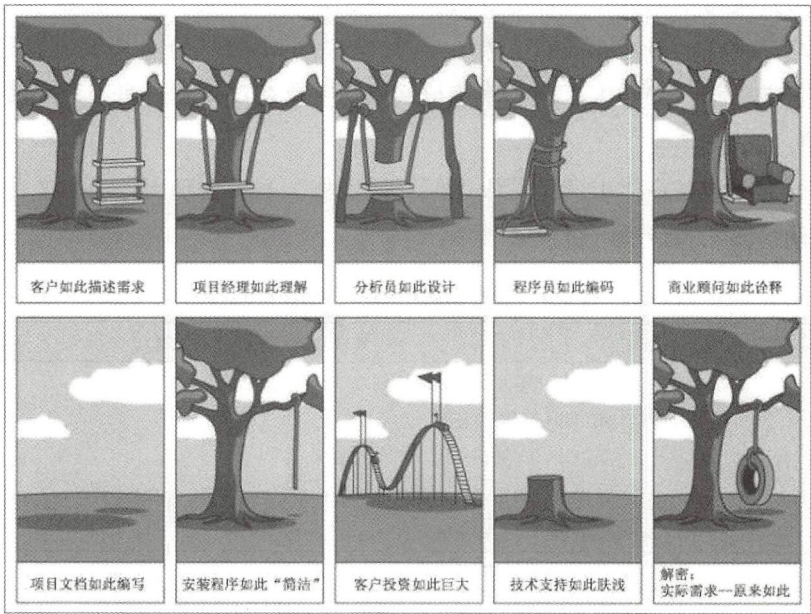


图 2-4 失败的需求调研

提示：笔者在开发某超市供应链交易平台之前，公司曾经组织整个团队学习供应商和超市交易的过程，后期到超市进行需求调研，对一些专有名词和交易过程都非常熟悉，甚至在有的操作过程中提出比用户的想法更简捷的建议，用户非常愉快地接受了。需求人员了解业务、熟悉业务，用户就会在潜意识中认可需求人员。

2.3.2 需求分析

需求是项目的灵魂，有了需求才有项目开展的可能。但是初期的用户需求并不能作为项目实施的依据，最多可以作为项目的指导性意见，因为初期的需求绝大部分是由业务部门发散收集的，不够具体，功能点不明确，没有逻辑，也没有具体流程，在实际的项目执行过程中并没有多大的参考意义。有了初期的用户需求雏形后，接下来就要进一步丰富和细化需求，每个功能点、计算逻辑、数据来源、处理流程、角色、权限、配置等，分析落地的可能性，然后进一步进行需求梳理、整合和丰富，进而加工成可以系统化的内容，进行产品选型，然后用技术实现。

需求分析的主要工作是对各种需求信息进行分析，将盘根错节的业务梳理清楚，定义规范的业务流程，并且让各方达成共识等，如图 2-5 所示。

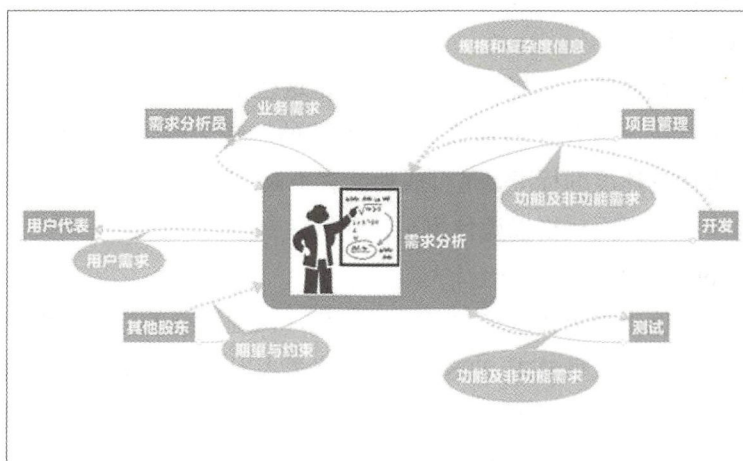


图 2-5 需求分析的主要工作

在需求开发阶段，需求分析是最关键的步骤，因为需求分析可以找到期望与约束，减少代码开发阶段的工作量，便于测试，保证产品质量。最简单的理解就是确保需求文档正

确地反映用户的真实意图。

在进行需求分析之前，应该先做可行性分析。可行性分析就是决定“做还是不做”。大家往往把这个关键的步骤省略。“做还是不做”“能不能做”“是否值得做”是可行性分析的3个关键点。可以召开一次可行性分析会议，请公司高层人员参加，明确这3个问题，得出最终结论，得到高层人员授权，再开始真正的需求分析。

提示：即使可行性分析是客观的、科学的，但决策仍有可能是错误的。因为决策者是人，人会冲动，有赌博心态。如果可行性分析表明做某件事的成功率是10%、失败率是90%，倘若这件事情的意义非常重大，决策者也许会一拍脑袋：“为了公司的长远发展，赔钱干！”那么项目组只能无奈的接受。

若需求范围界定不清，就会使项目缺乏计划，导致成本、开发周期失控。更严重的是，貌似需求人员组织大家召开需求评审会议，需求人员将会议上提出的意见写入需求跟踪矩阵，看起来就是一套遵循项目开发规范的流程，但是产生的问题是相当严重的。如果出现一个大的需求的错误，那么造成的结果就是推翻所有人的工作成果，再次重新来过。这是大部分公司经常犯的错误。

在需求文档中，通常采用两种方式：文字描述与建模分析。文字描述是第一重要的，它的标准称为 SMART，包括以下属性。

- Specific（明确的）

需求可以定义成快速的、友好的、直观的，但这些都不是明确的。每个人的理解是不一样的，比如词语“快速的”，1秒加载页面是快速的还是2秒加载页面是快速的？那么，评判这个产品是否达到速度的标准、是否成功就有争议。所以需求定义要使用具体的指标，例如响应时间为3秒、界面风格以蓝色为主。

- Measurable（可测性）

可测性在一定程度上要求需求能够明确、细化，保证同样的输入条件可以得到同样的输出结果。例如，性能测试里要求 TPS（每秒事务数）达到 1000，那么这里的事务是从哪个步骤开始到哪个步骤结束、涉及到多少数据量、什么内容、多少人在使用。再比如，功能测试里要求登录系统后需要输入用户名和密码，那么用户名或密码错误时会提示什么信息、最多限制多少次密码输错等。

- Agreed upon（达成共识的）

如果需求不被客户、项目人员、测试人员等各方所认可，那么在执行后续的设计开发

测试时会有所偏差，完成的产品不是预期所希望的样子。

- Realistic（现实的）

需求如果是不现实的，比如加载一个页面要求达到 1ms，那是无法达到的目标。这样的需求是无意义的。

- Time bound（时限性的）

定义需求什么时候可以确定，因为不可能一直等待需求的讨论评审。

建模主要是分析和解释作用。需求分析可以采用建立分析模型的方式，如事件列表、数据流图、实体关系图、数据流定义、数据字典、状态转换图、用例图、时序图、协作图、类图和状态图等。如图 2-6 所示是一个微信支付功能模块的建模实例。

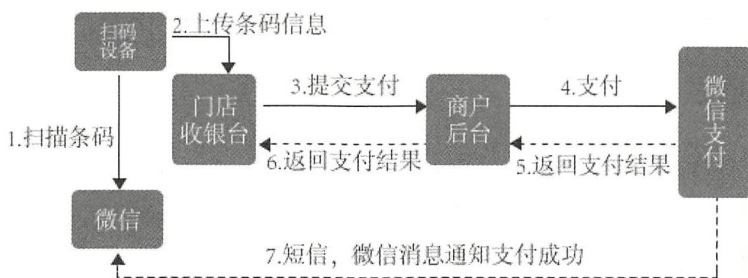


图 2-6 微信支付建模实例

需求分析阶段的关键点如下：

(1) 制作功能点控制表。把系统需求范围所涉及的功能点进行梳理，划分出功能点，对每个功能点进行编号，分配到具体的小组和项目成员；保证这些需求功能点涵盖全部需求范围。

(2) 没有对需求进行优先级划分，定义 90% 以上的需求都为高优先级，就会造成项目时间成本评估过高。在《需求规格说明书》里应该根据项目时间和资源成本合理定义需求的优先级。

(3) 提交需求分析报告，对于重点需求内容，要描述业务功能的流程和输入输出、业务规则、非功能性需求等内容。

提示：测试阶段修改一个 Bug 的成本是需求分析阶段修改成本的几十倍，甚至百倍。这个数值读者不用介意，总之意思很清晰，就是在需求阶段修改一个 Bug 的成本会比测试阶段修改一个 Bug 的成本低很多。

为了让读者更清楚在需求分析阶段如何开展工作，笔者梳理出如表 2-4 所示的需求分析过程中的输入输出和主要步骤，供读者参考。

表 2-4 需求分析阶段的输入输出和过程

输入项	《用户需求说明书》
主要步骤	第一步：问题分析。 第二步：问题评估和解决方案。 第三步：建模分析。 第三步：需求复审
输出项	《需求分析报告》
结束原则	软件需求人员编写《需求分析报告》，项目经理带领团队进行评审，配置管理工程师把《需求分析报告》纳入配置库

2.3.3 需求定义

需求定义的目标是根据需求调研和需求分析的结果，将《用户需求规格说明书》转换为《软件需求规格说明书》。系统设计人员将依据《软件需求规格说明书》开展系统设计工作。具体工作流程就是创建业务流程图和用例图，描述提议的解决方案。使用业务用例图、故事板、用户原型草图和其他资源支持需求定义。需求定义的主要工作如图 2-7 所示。在需求调研阶段是由项目经理主导工作，在需求分析和需求定义阶段是由需求人员进行具体的工作。

在图 2-7 中，what 的意思是“业务是什么”，how 的意思是“如何把业务用软件的方式实现”。

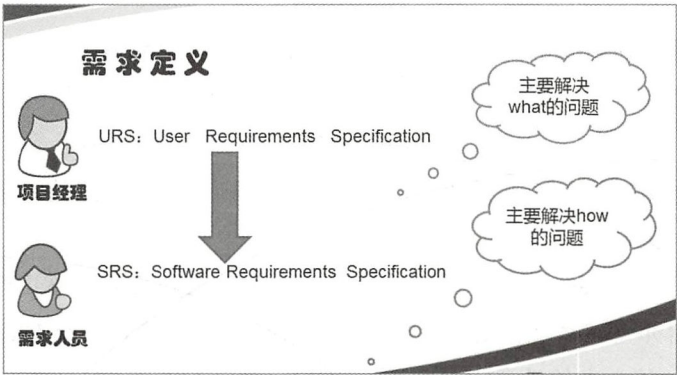


图 2-7 需求定义的主要工作

《用户需求规格说明书》转换为《软件需求规格说明书》，不是简单的一对一的转换，而是有明确的对应关系，千万不要简单地复制粘贴，具体的对应关系如表 2-5 所示。

表 2-5 对应关系表

用户需求规格说明书	软件需求规格说明书
产品的必要性及期望达成的目标； 产品价值体现，产品支持公司哪个策略	愿景，业务背景
产品干系人	涉众
产品监控方案	监控需求
术语定义	术语
必须满足的功能	范围
业务概述	业务分析
性能要求	非功能需求
风险和假设	风险

需求定义是整个需求开发阶段的最后一个步骤，同时也是最关键的步骤。如图 2-8 所示为《用户需求规格说明书》和《软件需求规格说明书》的成功转换实例。

<div>□ 1. 产品目标</div> <div>1.1 产品的必要性以及期望达成的目标</div> <div>1.2 产品价值体现</div> <div>1.3 产品目标和竞争对手有什么不同？怎样完成？</div> <div>1.4 这个产品支持公司哪一个策略？</div> <div>1.5 此产品在收益方面计划能够有何作用</div> <div>□ 2. 产品影响和成功标准</div> <div>2.1 产品对公司影响</div> <div>2.2 产品干系人</div> <div>2.3 产品所影响的具体业务</div> <div>2.4 产品所影响到的团队及其角色</div> <div>2.5 产品成功衡量标准</div> <div>2.6 产品监控方案</div> <div>2.7 产品需跟踪关键数据</div> <div>□ 3. 需求描述</div> <div>3.1 术语定义</div> <div>□ 3.2 业务概述</div> <div>□ 3.3 必须满足的功能</div> <div>3.4 质量要求</div> <div>3.5 性能要求</div> <div>3.6 其他非功能需求</div> <div>4. 产品运营</div> <div>5. 风险和假设</div> <div>6. Sign off</div>	<div>1. 文档简介</div> <div>2. 目的范围</div> <div>2.1 业务背景</div> <div>2.2 愿景</div> <div>2.3 范围</div> <div>3. 参考资料</div> <div>4. 术语</div> <div>5. 业务分析</div> <div>5.1 主要业务场景</div> <div>5.1.1 业务用例</div> <div>5.1.2 业务现状分析</div> <div>5.1.3 涉众分析</div> <div>5.1.4 业务解决方案</div> <div>6. 需求分析</div> <div>6.1 功能需求</div> <div>6.1.1 系统用例</div> <div>6.2 非功能性需求</div> <div>6.3 监控需求</div> <div>6.4 运营需求</div> <div>6.5 接口</div> <div>7. 影响的范围</div> <div>8. 软件限制条件</div> <div>8.1.1 解决方案的局限性</div> <div>8.1.2 外部系统依赖</div> <div>9. 风险</div> <div>10. 支持信息</div> <div>10.1 相关文档 SVN 地址</div>
--	---

图 2-8 转换实例

为了使读者更清楚需求定义阶段如何开展工作，笔者梳理出如表 2-6 所示的需求定义

过程中的输入输出和主要步骤，供读者参考。

表 2-6 需求定义过程中的输入输出和主要步骤

输入项	《用户需求规格说明书》
主要步骤	第一步：细化分析用户需求。 第二步：转换为《软件需求规格说明书》。 第三步：反复的修改和确认。 第四步：定版《软件需求规格说明书》
输出项	《软件需求规格说明书》
结束原则	软件需求人员编写《软件需求规格说明书》，项目经理带领团队进行评审，配置管理工程师把《软件需求规格说明书》纳入配置库

2.4 需求管理

需求管理是在客户与开发方之间建立对需求的共同理解，维护需求与其他工作成果的一致性，并控制需求的变更。需求管理包括需求确认、需求跟踪和需求变更管理。

- 需求确认是指开发方和客户共同对需求文档进行评审，双方对需求达成共识后做出书面承诺，使需求文档具有商业合同效果。
- 需求跟踪是指通过比较需求文档与后续工作成果之间的对应关系，建立与维护“需求跟踪矩阵”，确保产品依据需求文档进行开发。
- 需求变更管理是指依据“变更申请—审批—更改—重新确认”的流程处理需求的变更，防止需求变更失去控制而导致项目发生混乱。

2.4.1 需求确认

需求确认是指开发方和客户方共同对需求文档如《用户需求规格说明书》和《软件需求规格说明书》进行审计，双方对需求达成共识后做出承诺。《用户需求说明书》和《软件需求规格说明书》可以分开，也可以放在一起进行需求确认，视项目的具体情况而定。

需求确认包含两个重要工作，即“需求审计”和“需求承诺”。

需求审计就是最终确认，用户确认《软件需求规格说明》能正确并完整描述自己的需求。架构师确认根据《软件需求规格说明书》可以进行概要设计和详细设计的工作，开发



人员确认业务流程可以通过软件代码实现，测试人员确认业务流程清晰，可以编写测试用例，项目经理确认需求阶段的工作基本完成，可以进入下一个阶段。

需求承诺是指开发方和客户方的责任人对通过正式评审的《软件需求规格说明书》做出承诺，承诺的内容是双方同意后续的开发工作根据该《软件需求规格说明书》开展。如果需求发生变化，将按照“变更控制流程”执行。需求的变更将导致双方重新协商成本、资源和进度等。一定要让双方负责人在需求审计报告上签字。

为了使读者更清楚需求确认阶段如何开展工作，笔者梳理出如表 2-7 所示的需求确认过程中的输入输出和主要步骤，供读者参考。

表 2-7 需求确认过程中的输入输出和主要步骤

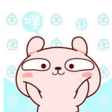
输入	需求文档，如《用户需求规格说明书》和《软件需求规格说明书》
主要步骤	第一步：非正式需求评审。 第二步：正式需求评审。 第三步：获取需求承诺
输出	《需求审计报告》和《需求承诺书》
结束准则	需求文档通过了正式审计，并且获得开发方和客户方的书面承诺

2.4.2 需求跟踪

需求跟踪矩阵（Requirement Tracking Matrix，RTM）是在需求变更、设计变更、代码变更、测试用例变更时，记录需求和需求变更、分析影响范围的最有效工具。如果不借助 RTM，发生上述变更时，往往会遗漏某些连锁变化。RTM 也是验证需求是否得到实现的有效工具。借助 RTM，可以跟踪每个需求的状态，如是否设计、是否实现、是否测试。RTM 可以帮助质量保证人员进行功能质量审计，也可以帮助项目经理进行需求管理。需求跟踪的目的是建立与维护“需求—设计—编程—测试”之间的一致性，确保所有的工作成果符合用户需求。如表 2-8 所示为需求跟踪矩阵的实例模板。

表 2-8 需求跟踪矩阵的实例模板

项目名称：				项目经理：				软件质量保证：	
序号	需求分析			系统设计		编码实现		系统测试	
	用户需求	功能需求	责任人	设计单元	责任人	代码单元	责任人	测试单元	责任人



需求跟踪有以下两种方式。

（1）正向跟踪。检查《软件需求规格说明书》中的每个需求是否都能在后继工作成果中找到对应点。

（2）逆向跟踪。检查设计文档、代码、测试用例等工作成果是否都能在《软件需求规格说明书》中找到出处。

为了使读者更清楚需求跟踪阶段如何开展工作，笔者梳理出如表 2-9 所示的需求确认过程中的输入输出和主要步骤，供读者参考。

表 2-9 需求确认过程中的输入输出和主要步骤

输入	需求文档、设计文档、代码、测试用例等
主要步骤	第一步：建立与维护需求跟踪矩阵。 第二步：查找不一致。 第三步：消除不一致
输出	需求跟踪报告（包含需求跟踪矩阵和问题处理）
结束准则	每个开发阶段的“需求跟踪矩阵”都已经建立。已经消除了需求文档与后续工作成果之间的不一致

2.4.3 需求变更管理

随着项目的进展，双方（包括开发方和客户方）对需求的了解越来越深入，原先的需求文档可能存在这样或那样的错误或不足，因此要变更需求。又或者是市场发生了变化，原先的需求文档跟不上当前的市场需求，因此也要变更需求。

需求变更管理的目的是，如果需求变更带来的好处大于坏处，那么允许变更，但必须按照已定义的变更规程执行，以免变更失去控制；如果需求变更带来的坏处大于好处，那么拒绝变更。

需求变更管理的关键点如下。

（1）假设需求以失控的状态进入软件过程，从源头上失去了项目的质量保证。这种情况一般是需求人员讲解完《需求规格说明书》后，项目经理马上就安排架构师和开发人员进入设计和实施阶段。实际上，需求没有被完全掌握，也没有进行分析，甚至没有评审，大家就自然而然地接受了。在研发人员发现问题后，先改代码，再更改需求，并且称之为需求变更，然后项目团队开始申请延期，所以一定要避免这种情况。



(2) 需求变更何时最容易出现？根据笔者的经验，80%的需求变更发生在测试阶段，15%的变更发生在需求确认至代码完成期间，剩余的5%发生在线上试运行阶段，所以项目经理或产品经理也无法阻止实现需求后仍然有需求变更的情况出现。

(3) 需求变更超过项目总体的30%，那么项目就需要重新立项，因为目前的时间和人员肯定是无法完成工作的，加班也不行。这里笔者郑重呼吁：加班是低效的，过往的项目都证明了这一点，8小时工作时间如果充分利用，可以提高积极性，否则会造成项目越来越失控，人员增加，成本增加，工时增加，最终的结果就是一种无限期的延迟下去。

(4) 需求变更失控，不停地要求增加修改功能，使项目团队处于被动接受的状态中，项目组成为“救火队”。这种情况下要规范变更管理流程，考虑预留资源，对需求实施基线管理，用良好的需求管理工具来支撑。

为了使读者更清楚需求变更阶段如何开展工作，笔者梳理出如表2-10所示的需求变更管理过程中的输入输出和主要步骤，供读者参考。

表 2-10 需求变更管理过程中的输入输出和主要步骤

输入	原需求文档（指已经通过了评审并获得书面承诺的需求文档）
主要步骤	第一步：变更申请。 第二步：审批。 第三步：更改需求文档。 第四步：重新确认需求。 第五步：结束变更
输出	《需求变更控制报告》
结束准则	新的需求文档已经完成并被确认

2.5 需求说明书

通常认为只有两种需求说明书，即《用户需求规格说明书》和《软件需求规格说明书》。这种说法其实漏掉了最关键的一项内容，即《业务需求规格说明书》。

从《业务需求规格说明书》到《用户需求规格说明书》，再从《用户需求规格说明书》到《软件需求规格说明书》，其实就是一个逐步细化的过程，将行业术语先转换为用户语言，再转换为计算机语言。



2.5.1 《业务需求规格说明书》

需求讨论集中于业务需求和任务，因此要使用术语。客户可以使用相关术语（如采价、印花商品等采购术语）为需求人员讲解，不一定要懂得计算机行业的术语。因此，需求人员要了解客户的业务及目标。业务需求来源于两个部门，即商务部门和市场部门，其产品的文档分别如下。

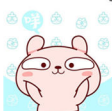
- 商业需求文档（Business Requirement Document）。它是一种商业的需求描述，里面要体现产品的市场分析、规划、投入、盈利预测等信息，是决策层分析、决策是否开发此产品的依据。商业需求文档更像是商业计划书，是需求阶段最早提供的文档，文档一般不长，也可以是以 PPT 的方式展示。
- 市场需求文档（Market Requirement Document）。市场需求文档是站在市场、用户的角度，更多地描述用户、购买者和客户的需求，它是承上启下的文档，对后续的软件需求文档编写有一定的指导作用。文档中多会以原型的形式将产品具体化，便于文档的解释说明。

《业务需求规格说明书》就是商务部门和市场部门根据行业发展、市场需求，从公司实际情况出发，提出的开发某类产品的建议，通常是战略性或者前瞻性的规划。是否根据提交的《业务需求规格说明书》进行研发工作，这是由公司高层决定的。

例如，在 2005 年电子商务发展初期，商户亟需解决网上收款问题。商户接入各银行收单接口的技术门槛高，运营成本大。若研发电子支付—人民币网关产品，就可以解决互联网交易的收款问题，降低商户接入银行的技术及运营投入。同时由于公司具有专业化的支付工具整合的能力、专业的交易风险控制能力且银行接口齐全，所以公司可以开发人民币网关产品。如表 2-11 所示为行业内竞争对手的优劣势。

表 2-11 行业内竞争对手分析

竞争者	优势	劣势
某联	资本能力强； 研发实力强； 银行接口多	市场营销能力一般； 定位支付转接； 产品的客户体验较差
某宝	现有交易量规模大，用户规模大； 产品在便捷性上处于行业领先； 产品研发和推广的投入大，速度快	关注普通用户，没有关注商户



续表

竞争者	优势	劣势
某通	用户规模大; 资本能力强; 产品研发的投入较大	无快捷支付产品; 用户信息不安全, 与商户处于竞争地位
某钱	行业专注度高	无快捷支付产品; 产品的客户体验一般

2.5.2 《用户需求规格说明书》

《用户需求规格说明书》描述的只是业务场景, 可根据业务场景画出业务实现流程图, 并用业务场景来验证《软件需求规格说明书》中的业务实现是否正确。

例如, 假如公司高层同意研发支付系统, 那么需求人员就要按照《业务需求规格说明书》中定义的基本方向, 总结出关键点, 编写《用户需求规格说明书》。《业务需求规格说明书》仅仅是参考和指导性意见, 最终要编写的《用户需求规格说明书》必须经过多轮用户调研和详细分析。

接下来的工作就是梳理关于支付的各方面知识, 列出支付系统的定义和流程。

支付系统的基本定义是: 用户在互联网上购买商家商品或服务时, 由商家将用户及商品相关信息提交给支付系统, 用户在支付系统可选择任意一种支付方式, 输入相应的身份认证信息完成支付后, 最终由支付系统将资金结算给商户。

简单的支付流程如下。

- (1) 消费者浏览购物网站, 购买商品, 提交订单。
- (2) 选择支付, 跳转到收银台界面, 选择相应的支付方式 (如第三方支付平台)。
- (3) 消费者在打开的第三方支付平台中选择支付的银行卡 (如消费者的中国银行卡)。
- (4) 消费者在第三方支付平台已经注册为用户, 并且绑定中国银行卡。
- (5) 第三方支付平台转发消费者银行卡信息到相关银行。
- (6) 中国银行处理支付请求, 经过验证后授权第三方支付。
- (7) 中国银行发送短信通知消费者, 银行卡有消费记录。
- (8) 第三方支付平台将应答发送给购物网站。
- (9) 购物网站确认交易成功后向消费者寄送货物。
- (10) 第三方支付平台根据协议向购物网站清算和结算。



质量全面管控：从项目管理到容灾测试

支付流程示意图如图 2-9 所示。

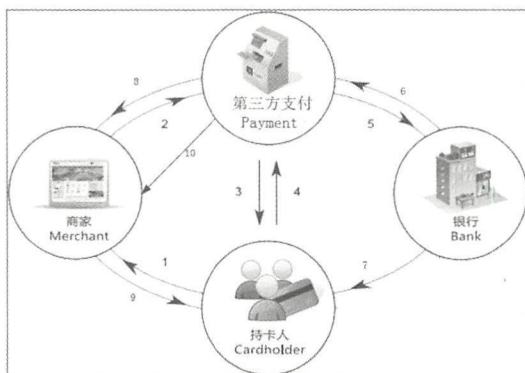


图 2-9 支付流程

2.5.3 《软件需求规格说明书》

《软件需求规格说明书》重点是确认产品的开发意图、应用目标及作用范围，介绍产品所具有的主要功能。可以用列表的方式给出，也可以用图形表示它们之间的联系，例如数据流程图的顶层图或类图等。

《用户需求规格说明书》与《软件需求规格说明书》的主要区别与联系如下。

(1) 前者主要采用自然语言和行业术语来表达用户需求，其内容相对于后者而言比较简略，不够详细。

(2) 后者是前者的细化，更多地采用计算机语言和图形符号来描述需求，软件需求是软件系统设计的直接依据。

(3) 两者之间可能并不存在一一对映关系，因为软件公司会根据产品发展战略和企业当前状况适当地调整产品需求。例如，用户需求可能被分配到软件的数个版本中。软件开发人员应当依据《软件需求规格说明书》来开发当前产品。

例如，下面是支付平台收单业务的软件需求，流程图如图 2-10 所示。

(1) 收银台根据业务类型、用户和商户支持交易的类型，显示可用的支付方式。用户选择相应的支付方式，调用交易核心。

(2) 交易核心调用风控系统，校验交易规则和风险。

(3) 交易核心根据支付方式和交易类型，生成支付指令，调用支付系统的支付核心进



行支付。

- (4) 支付核心调用渠道和账务系统完成支付，将支付结果同步或者异步返回给上游。
- (5) 交易核心将结果通知风控系统和清结算系统。

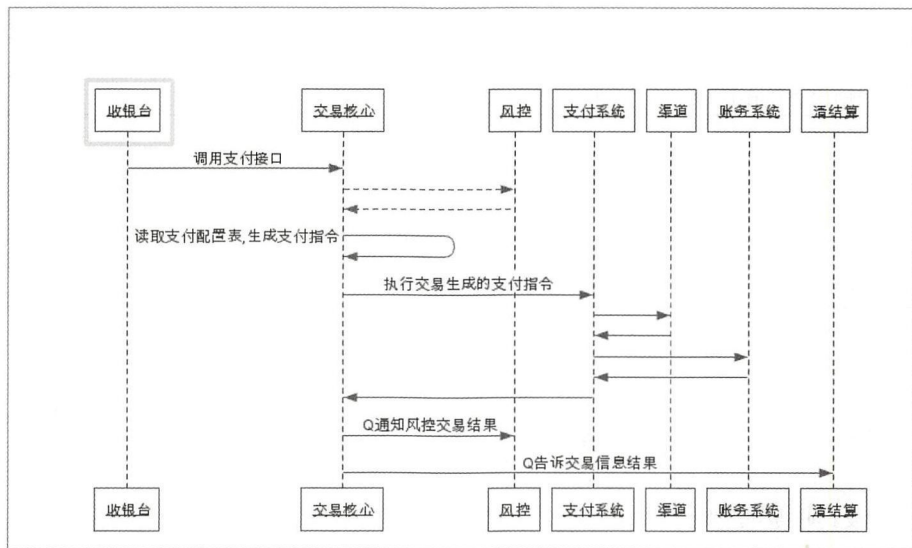


图 2-10 软件需求流程图

2.6 测试需求

在需求开发和管理流程中，测试人员也需要参与到需求评审中，参与会议讨论，对需求进行分析和验证。在需求阶段，测试需求非常考验测试人员的能力，单凭测试人员的经验是远远不够的。如表 2-12 所示是测试人员在需求阶段测试需求的关键点。

表 2-12 测试需求的关键点

序号	内容	正确/错误	问题描述
1	是否描述系统的所有输入，包括输入源、准确性、取值范围和出现频率		
2	是否描述系统的所有输出，包括输出的目标、准确性、取值范围、出现频率和格式		
3	是否描述所有（主要）的报表格式		
4	是否描述所有硬件和软件的外部接口		



质量全面管控：从项目管理到容灾测试

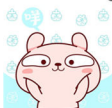
续表

序号	内容	正确/错误	问题描述
5	是否描述所有的通信接口，包括握手协议、差错检测和通信协议		
6	从用户的角度来看，是否描述了对所有必要操作的预计响应时间		
7	是否对时间方面的问题进行了考虑，如处理时间、数据传输和系统的吞吐量		
8	是否描述用户想要完成的所有（主要）任务		
9	是否每个任务都描述了所使用的数据及产生的数据		
10	是否描述了安全级别		
11	是否描述了系统的可靠性，包括软件产生故障的后果、故障后重要数据的保护、错误检测和恢复		
12	是否描述了可接受的折中原则，如健壮性和正确性之间的选择		
13	是否详细说明了（最大）内存容量和（最大）存储容量		
14	是否详细说明了（最大）存储容量		
15	是否详细说明了系统的可维护性，包括适应操作环境变化的能力、与其他软件的接口、精确性、性能和附加的可以预知的性能		
16	有些信息只有到开发时才能获得，是否对这些信息不完全的领域进行了描述		
17	是否对需求的某些部分感到不满意？是否有些部分不可能实现，但为了取悦客户或上司而放在需求之中		
18	是否用用户语言、站在用户的角度来写需求？用户这样认为吗		
19	是否所有的需求都避免与其他的需求发生冲突		
20	需求是否避免了对设计的详细说明		
21	对需求的描述是否一致？是否有的需求说明很详细，有的需求说明很简短		
22	需求是否足够清晰，以至可以转交给一个独立小组来实现并能够被理解		
23	每个条款都是描述问题及解决问题吗？每个条款都能被追溯到问题的源泉吗		
24	每个需求都是可测试的吗？是否可以通过独立的测试来决定需求是否被满足		

2.7 需求管理工具

在选择需求管理及变更管理工具时，一般认为应该至少具备如下几个特征。

- 最基本的要求是工具需提供信息记录功能，可以起到备忘录及交流的作用，信息要更透明，做到有理有据。



- 考虑到变更度量的复杂性，尤其是要适应不同项目特征、目的和统计理论，工具应提供一个灵活、便捷的查询统计机制，方便针对各种度量数据进行报表定制和浏览。
- 需求管理系统应和其他过程管理系统相配合，不能成为信息孤岛。

2.7.1 Rational RequisitePro

IBM Rational RequisitePro 是一个需求和用例管理工具，能够帮助项目团队改进项目目标的沟通，增强协作开发，降低项目风险，以及在部署前提高应用程序的质量。通过与 Microsoft Word 高级集成，为需求的定义和组织提供熟悉的环境。还提供数据库与 Word 文档的实时同步功能，为需求的组织、集成和分析提供方便。

Rational RequisitePro 支持需求详细属性的定制和过滤，以最大化各个需求的信息价值；提供了详细的可跟踪性视图，通过这些视图可以显示需求间的父子关系，以及需求之间的相互影响关系；通过导出的 XML 格式的项目基线，可以比较项目间的差异；可以与 IBM Software Development Platform 中的许多工具进行集成，以改善需求的可访问性和沟通。Rational RequisitePro 界面如图 2-11 所示。

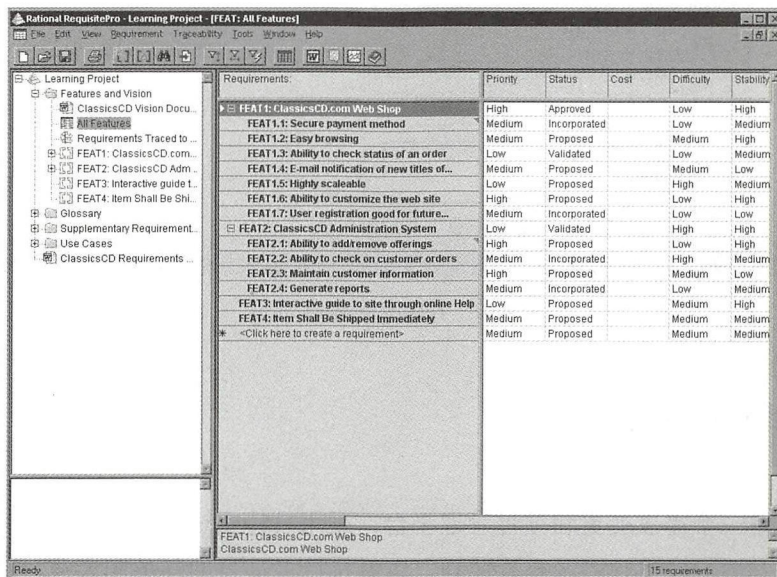


图 2-11 Rational RequisitePro 界面

质量全面管控：从项目管理到容灾测试

2.7.2 TelelogicDoors

TelelogicDoors 是基于整个公司的需求管理系统，用来捕捉、链接、跟踪、分析及管理信息，以确保项目与特定的需求及标准保持一致。它通过需求库实现更高生产率的建设性的协作，并且根据特定的需求定义可交付物，提供可视化的验证方法，从而达到质量标准。

TelelogicDoors 企业需求管理套件是仅有的面向管理者、开发者与最终用户及整个生命周期的综合需求管理套件。不同于那些只能通过一种方式工作的解决方案，它提供多种工具与方法对需求进行管理，可以灵活地融合到公司的管理过程中。TelelogicDoors 使整个企业能够有效地沟通，从而减少失败的风险。它通过统一的需求知识库，提供对结果是否满足需求的可视化验证，从而达到质量目标，并能够进行结构化的协同作业，使生产率得到提高。TelelogicDoors 界面如图 2-12 所示。

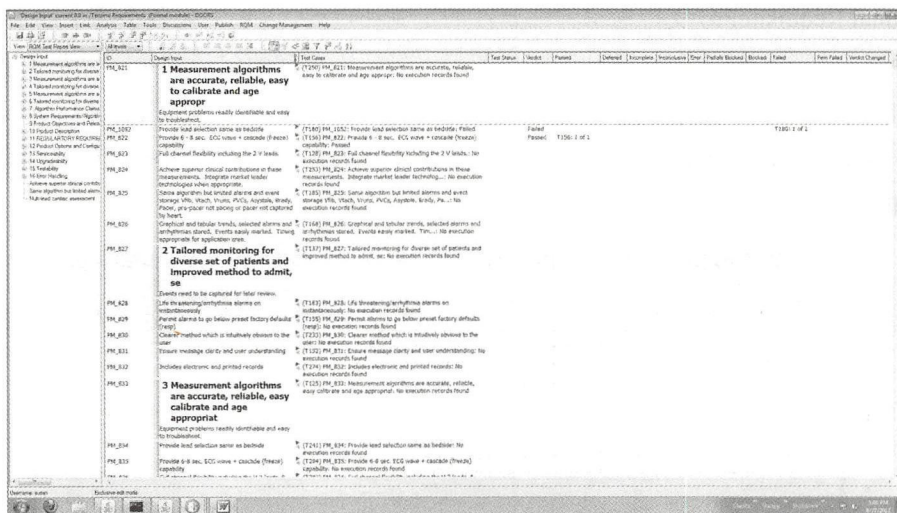


图 2-12 TelelogicDoors 界面

2.8 Plandora 实战

Plandora 是一个需求变更和项目管理平台，采用甘特图进行任务计划，将项目周期存储在知识库中，还可以进行资源容量计划，查看执行多项目的任务和人员，跟踪里程碑和管理项目文档，跟踪管理财务，导入导出内容。

Plandora 的下载网址是 <http://www.plandora.org/>, 目前最新版本为 Plandora 1.14.0。Plandora 演示系统可提前体验, 网址是 <http://www.plandora.org/online.htm>。

2.8.1 搭建 Plandora

Plandora 部署需要 Web 服务器和数据库, 建议使用 Tomcat 和 MySQL, 也可以使用其他 Web 服务器和数据库。配置 Plandora 的步骤如下。

(1) 第一次安装时, 在 MySQL 上创建一个 DB schema, 命名为“plandora”, 然后运行 zip 文件中的 Mysql.SQL, 创建基础数据。

(2) 复制 mysql-connector-java-5.1.10-bin.jar 到 Tomcat 的 lib 目录下。这个 JAR 文件用来连接 MySQL 数据库。

(3) 配置数据源。

- Tomcat5.0.x 需要复制安装包中的 server.xml 到 Tomcat 的 conf 目录中。
- Tomcat5.3.x 及以上版本需要复制安装包中的 context.xml 到 Tomcat 的 conf 目录中。

例如, 在上面提到的 context.xml 中配置正确的数据源, 修改数据库的用户名、密码和连接串。

```
<Resource name="jdbc/plandora" auth="Container"
    type="javax.SQL.DataSource"
    username="root"
    password="123456"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://127.0.0.1:3306/plandora"
    maxActive="80" maxIdle="50"/>
```

(4) 上传 pandora.war 到 Tomcat 的 webapps 目录下。

(5) 启动 Tomcat, Windows 平台下执行 startup.bat, Linux 平台下执行 startup.sh。启动 Plandora, 打开浏览器, 访问路径为 <http://127.0.0.1:8080/pandora/do/login?operation=prepareLogin>。

登录界面配置成功, 如图 2-13 所示。初始用户名为 root, 默认无密码。管理员登录成功后, 可以添加用户。

提示: 必须修改用户名和密码, 但勿删除 root 用户, 这是系统的管理员, 系统有了它才能正常运行。

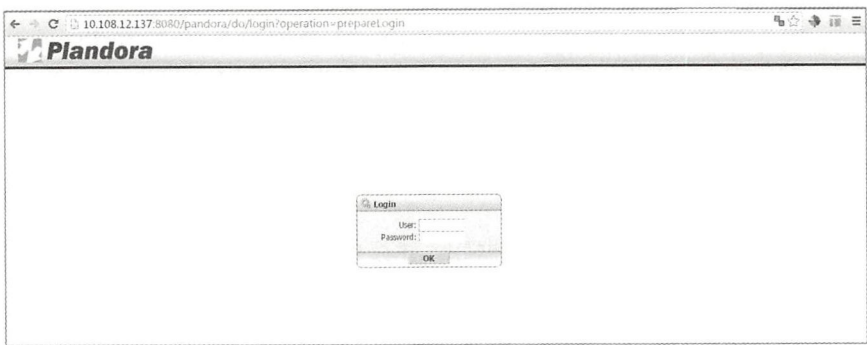


图 2-13 Plandora 登录界面

2.8.2 管理员配置

Plandora 被设计成前端和后端。前端是应用程序用户可以执行的项目，如甘特图、任务追踪、资源分配、财务管理、风险和报表查看等功能。后端则是管理员为此应用创建的公司组织和项目及用户的配置。

1. 创建公司信息

使用管理员账号 root 登录系统，如图 2-14 所示，可以设置 Company（公司）、User（用户）和 Project（项目）等信息。

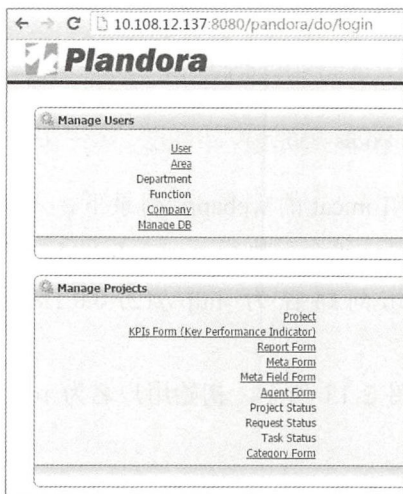


图 2-14 管理员界面

选择“Company”链接,在打开的如图 2-15 所示的页面中,输入 Name(姓名)、Full Name(全名)、ID Number(公司代码)和 Address(地址)等信息。单击“Save New”按钮保存设置,新建的公司信息显示在“Company List”区域中。

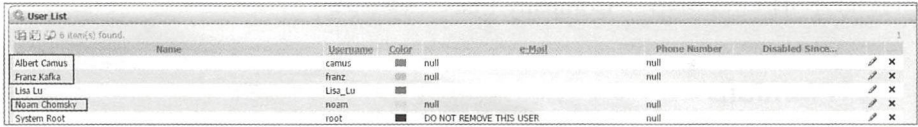
图 2-15 公司设置页面

2. 创建用户

在公司设置页面中单击“Main Form”按钮,返回主界面。选择“User”链接,打开新建用户页面,输入 Name(姓名)、Username(用户名)、Color(颜色)、Password(密码)和 Confirmation(确认密码)等信息,也可以上传照片,如图 2-16 所示。完成后单击“Save New”按钮保存新用户信息。

图 2-16 新建用户页面

新建的用户信息会显示在“User List”区域，如图 2-17 所示。如果要删除用户，必须把给用户分配的项目关闭后才可以删除，单击条目右边的 × 按钮即可。

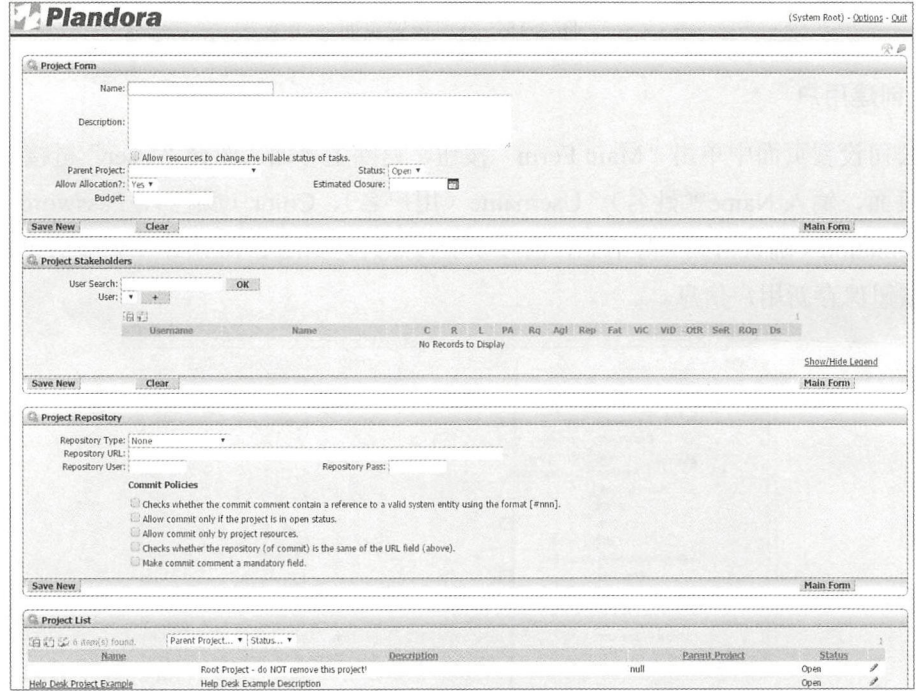


Name	Username	Color	e-Mail	Phone Number	Disabled Since...
Albert Camus	camus		null	null	x
Franz Kafka	franz		null	null	x
Lisa Lu	Lisa_Lu		null	null	x
Noam Chomsky	noam		null	null	x
System Root	root		DO NOT REMOVE THIS USER	null	x

图 2-17 用户列表

3. 创建项目

在新建用户页面单击“Main Form”按钮，返回主界面。选择“Project”链接，打开新建项目页面，如图 2-18 所示。可以输入 Name（项目名称）、Description（描述）、ParentProject（父项目）及其他信息。



Project Form

Name:

Description:

Parent Project: Status:

Allow Allocation?: ☐ Yes ☐ No Estimated Closure:

Budget:

Project Stakeholders

User Search:

User:

Username	Name	C	R	L	PA	Rg	Apl	Rep	Fat	VIC	VID	OCR	Self	ROP	DS
No Records to Display															

Project Repository

Repository Type:

Repository URL:

Repository User: Repository Pass:

Commit Policies

☐ Checks whether the commit comment contain a reference to a valid system entity using the format {#nnn}.

☐ Allow commit only if the project is in open status.

☐ Allow commit only by project resources.

☐ Checks whether the repository (of commit) is the same of the URL field (above).


☐ Make commit comment a mandatory field.

Project List

Name	Description	Parent Project	Status
Root Project - do NOT remove this project!	Help Desk Example Description	null	Open
Help Desk Project Example			Open

图 2-18 新建项目页面

需要注意，项目必须要分配给用户才能创建。在“Project Stakeholders”区域的“User Search”文本框中输入要查询的用户，例如“Lisa”，单击“OK”按钮，此时“User”下拉

列表框中会显示被查询到的用户，在下拉列表框中选择好用户，单击  按钮，即可将项目分配给用户，如图 2-19 所示。

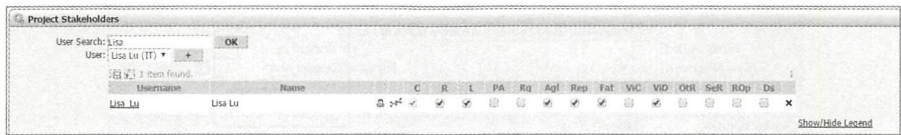



图 2-19 分配项目

分配的资源要授予合适的权限，单击右下角的“Show/Hide Legend”链接可以查看或隐藏各个列的权限内容，现摘录如下：

- C——项目客户（Customer）；
- R——项目资源（Resource）；
- L——项目领导（Leader）；
- PA——允许客户创建预批准请求（Request）；
- Rq——允许资源查看客户名字（Requester）；
- Agl——允许资源访问敏捷板（Agile Board）；
- Rep——允许资源访问项目仓库（Project Repository）；
- Fat——允许资源访问项目发票（Project Invoices）和开销（Costs）；
- ViC——允许客户查看资源汇报的技术备注（Technical Comments）和编辑请求关系（relationship of requests）；
- ViD——允许客户查看有关请求的讨论话题（Discussion Topics related To Requests）；
- OtR——允许客户查看项目中其他客户提交的请求（requests）；
- SeR——允许客户代替其他人管理请求；
- ROp——允许客户重开已关闭的请求；
- Ds——禁止领导/资源/客户参与项目。

也可以在项目选择添加项目仓库，如 SVN 地址或系统数据库地址，单击“Save New”按钮保存新项目信息。

在“Project List”区域中会显示新增的项目，要修改项目只需要选中一条记录，单击  按钮即可。删除已关闭的项目时，需要先进入项目页面，然后在“Status”下拉列表框中选择“Closed”或“Aborted”，如图 2-20 所示。

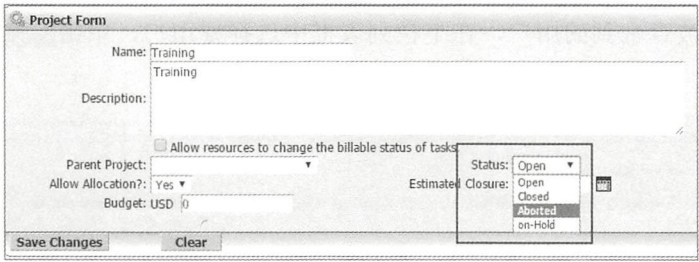


图 2-20 修改项目状态

2.8.3 前端用户

管理员配置完项目和用户后，这个项目才可以被前端用户所使用。前端用户登录 Plandora，可以看到主界面中显示 “My Tasks”、“My Requests” 和 “My Project” 等。如果是 “PL” 角色的用户，还可以看到 “My Teams”。

在主界面中可以看到用户在项目中扮演的角色和权限，如图 2-21 所示的用户是 “Leader”，能够访问大部分的功能模块，如 Risks、Books、Surveys、Invoices、Costs、Knowledge、Reports、Score Card、Agile Board、Tasks、Requests、Gantt Chart 和 Resource Capacity Planning。

用户可以给项目创建任务和请求，然后跟踪花费的时间，产生报告和评估进度。

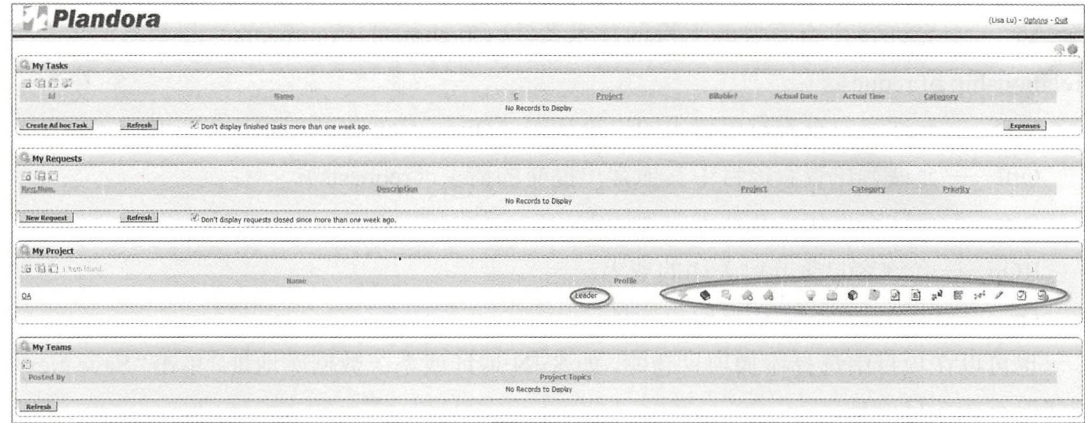


图 2-21 前端用户主页

Plandora 系统中的 Request（请求）等同于一个需求，所有的需求信息都显示在 Request 界面。当然也可以根据需要将 Request 名称修改为 Requirement，如图 2-22 所示。

Requirement Form

Requester: oscar Request Num.: 1024 *Suggest Date: []

Project: Service Desk
Category: Failure-Hardware-Other
Priority: Medium

Description: Failed hard disk.

Meta Field Example: This is a meta field example...

* The Suggested Date only express the deadline intended by requester and can be used, or not, by project leader into planning.

Discussion List

Posted By: [] at Nov 20, 2010 5:30:51 PM
Dunn: [] Explain exactly what's happened?

Save Changes Clear Main Form

Requirement List

12 item(s) found, displaying from 1 to 4

Req.No.	Description	Requester	Project	Priority	Category	Status	Actions
690	As the marketing department I want to produce a product brochure to hand-out at our internal conference	oscar	Foo Project	Medium	User Story	Waiting Approve	[] [] []
677	As a manager I want to be able to view a report on all new newsletter subscribers for a given...	oscar	Foo Project	Medium	User Story	Waiting Approve	[] [] []
645	As the system I must send out a confirmation email when a subscriber successfully registers for the newsletter.	oscar	Foo Project	Low	User Story	Waiting Approve	[] [] []
624	Failed hard disk.	oscar	Service Desk	Medium	Failure-Hardware-Other	Waiting Approve	[] [] []

Refresh [] Don't display Requirements closed since more than one week ago.

图 2-22 修改需求名称

1. 新增请求

在主界面的“**My Requests**”区域中，单击“**New Request**”按钮，打开新增需求页面，如图 2-23 所示。输入 Project、Category、Priority、Suggest Date、Estimated Time、Allocated Resource 和 Description。

提示：

(1) 勾选“This is a ‘pre approve request’”复选框，会自动添加一个关联的 task (任务)。

(2) 下拉列表框“Category”和“Priority”的值可以在管理员页面的“Category Form”链接中设置。可根据 Project 和 Category 添加修改值。

Request Form

Requester: Lisa_Lu Request Num.: [] *Suggest Date: 04/14/2016

Project: QA
Category: Enhancement
Priority: High

Description: Project Enhancement to allow provider update tickets.

☒ This is a 'pre approve request' (After saved, a task must be created to the selected resource automatically).

* The Suggested Date only express the deadline intended by requester and can be used, or not, by project leader into planning.

Save New Clear Main Form

Request List




3 item(s) found, displaying from 1 to 3

Req.No.	Description	Requester	Project	Priority	Category	Actions
211	Story 1	Lisa_Lu	Training	to be defined	Other	[] [] []
206	Request to test abcd	Tony	QA	Lower	Other	[] [] []

Refresh [] Don't display requests closed since more than one week ago.

图 2-23 新增需求页面



新增的需求会显示在“Request List”区域中，对于需求可添加附件（单击按钮）或者修改请求（单击按钮）。单击按钮，打开“Customer Request Adjustment”页面，如图 2-24 所示，可以修改需求中的 Project（项目）、Category（类别）和 Priority（优先级），也可以修改附件和关联任务，还可以添加讨论。

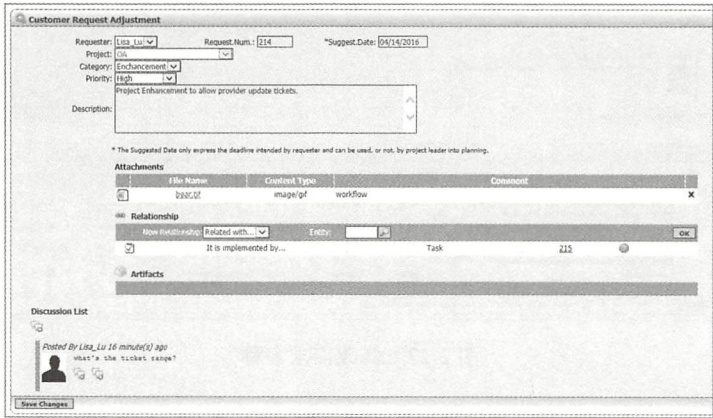


图 2-24 需求调整页面

2. 新建任务

在主界面的“My Tasks”区域中，单击“New Adhoc Task”按钮，打开“Task Tracking Form”（新建任务）页面，如图 2-25 所示，可以设置 Project、Category、Name、Description、Status 和 Estimated Time 等。类似于需求页面，Category 选项可以在管理员页面的“Category Form”链接中设置。

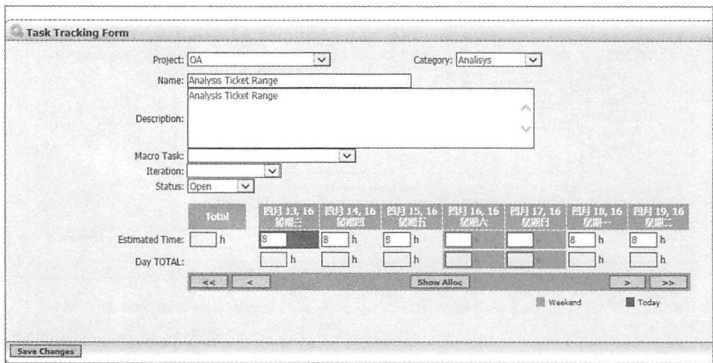


图 2-25 新建任务页面



新建任务完成后，会显示在“Tasks of Resource”区域中，如图 2-26 所示。可对任务进行删除、修改，也可以添加附件。

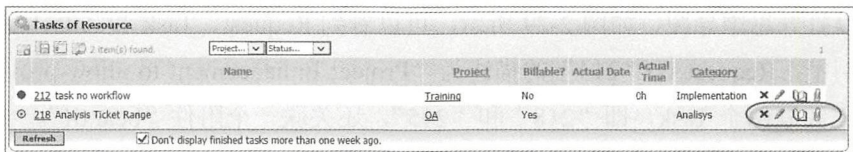



图 2-26 任务列表

单击  按钮，打开“Task Tracking Form”页面。可以修改 Status（状态），设置 Actual Time（实际时间），也可以修改附件和关联任务，还可以添加讨论，如图 2-27 所示。

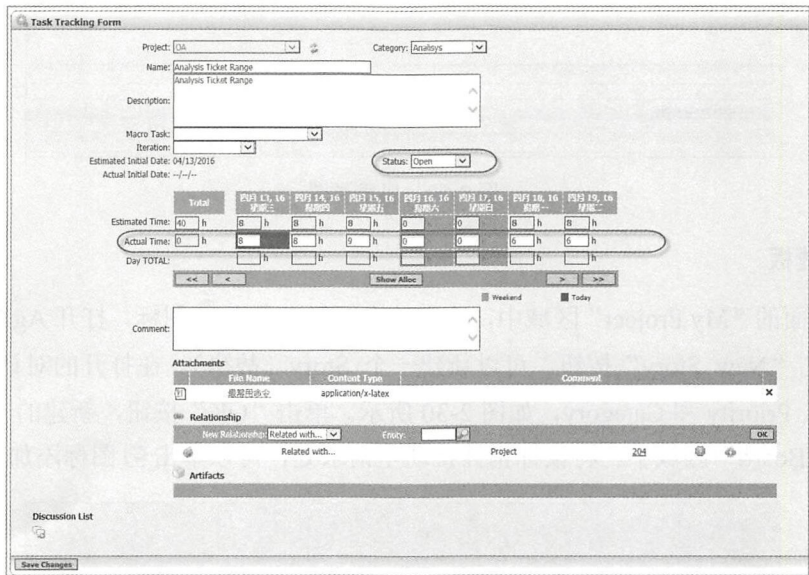


图 2-27 修改任务

修改 Actual Time（实际时间）后，系统会计算总的实际时间，如图 2-28 所示。

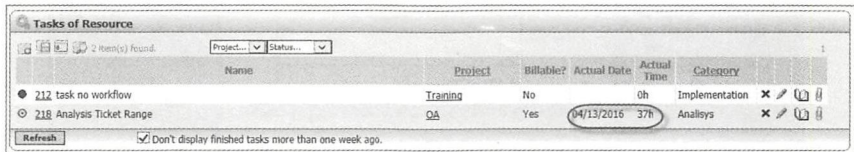


图 2-28 任务实际时间



质量全面管控：从项目管理到容灾测试

3. 思维导图

在主界面“My Tasks”和“My Requests”区域中，单击“Task ID”或“Request ID”链接，可以打开思维导图，如图 2-29 所示，可以看到 Request、Task 和 Attachment 之间的关联。示例中，Request ID “214” 的描述为“Project Enhancement to allow provider update tickets”，已关联两个 Task，即“218”和“215”，还关联一个附件“bear.tif”。

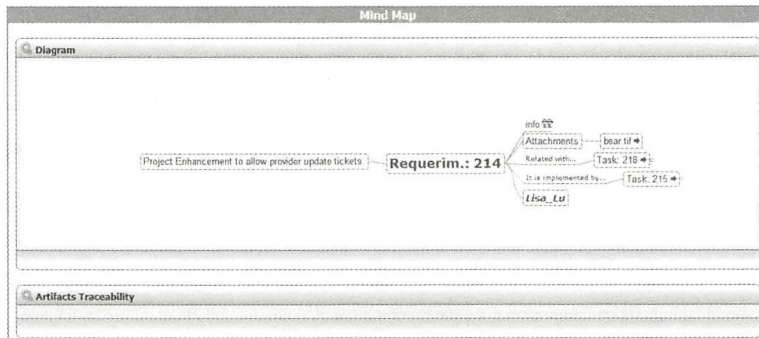




图 2-29 思维导图

4. 敏捷板

在主界面的“My Project”区域中，选定一个项目，单击  图标，打开 Agile Board（敏捷板）。单击“New Story”按钮，可以新建一个 Story（故事）。在打开的对话框中可以设置 Description、Priority 和 Category，如图 2-30 所示。单击“OK”按钮，新建的 Story 就会显示在“Task Board”区域中。将鼠标指针移动至请求处，可以单击  图标添加任务。

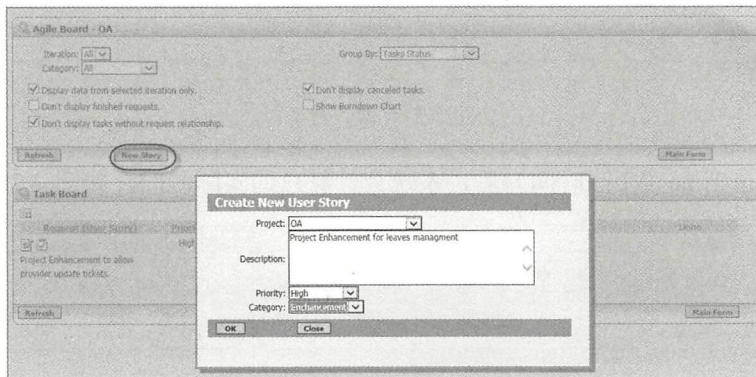


图 2-30 新建 Story



5. 风险管理

在主界面的“**My Project**”区域中，选定一个项目，单击⚡图标，打开“**Risk Form**”（风险表单）页面，可以设置 **Name**、**Description**、**Type**（**Threat/Opportunity**）、**Category**、**Probability**、**Impact**、**Tread** 和 **Status** 等，如图 2-31 所示，然后单击“**Save New**”按钮。

图 2-31 风险表单页面

新增的风险表单可以进行修改，选中一个风险表单，将“**Status**”下拉列表框中的选项设置为“**Materialized**”，单击“**Save Changes**”按钮保存，会弹出如图 2-32 所示的对话框。

图 2-32 修改确认对话框


单击“**Yes, Save Risk and Issue**”按钮，会新建一个 **Occurrence**（事件），如图 2-33 所示。**Occurrence** 也可以通过在主界面中单击项目的⚡图标进行提交。

图 2-33 新建事件



质量全面管控：从项目管理到容灾测试

6. 查看图表

在主界面中，单击右上角的  图标，可以设置图表，如图 2-34 所示。

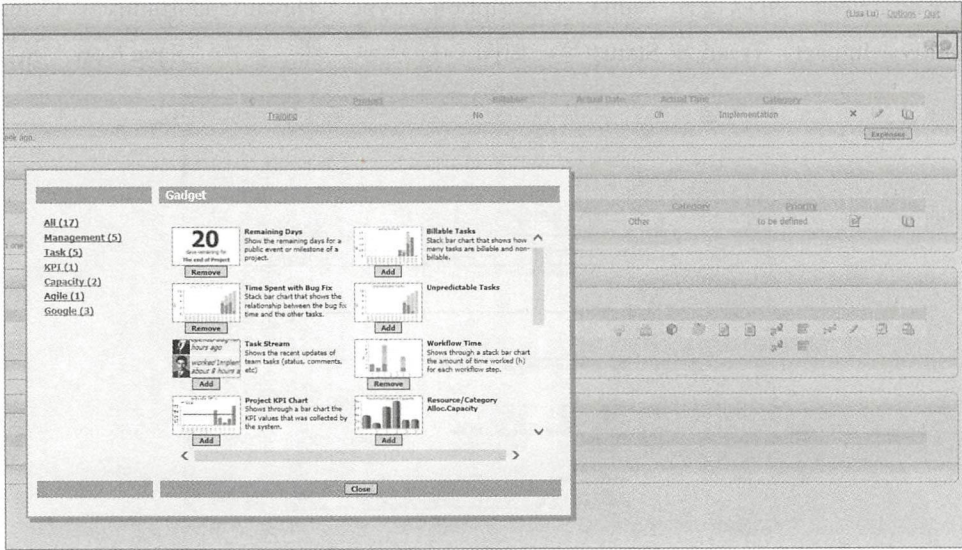


图 2-34 图表设置

2.8.4 需求变更实例

在项目运行中，有时会有需求的变更。例如，如图 2-35 所示是一个 CRM 的开发项目，里面有 franz 和 Lisa_Lu 负责的项目，camus 是开发人员，noam 是产品经理。现在产品经理需要提交一个变更需求，“要求允许联系人信息可以同步到云”。

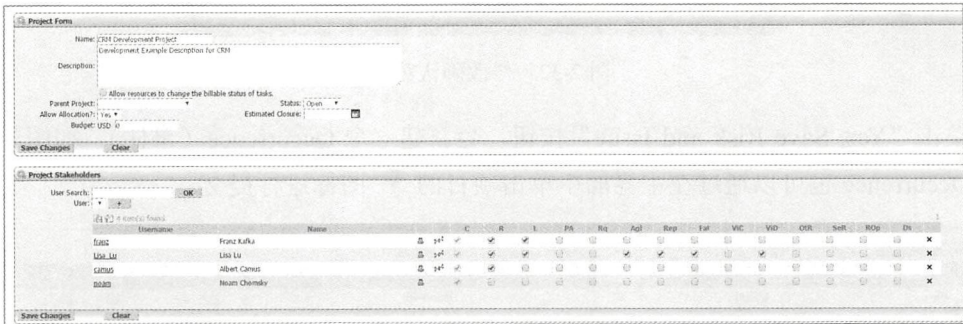


图 2-35 项目信息及资源



1. 新增需求变更

产品经理 noam 登录系统，在主界面中单击“New Request”，打开需求页面。设置项目为“CRM Development Project”、类别为“Change”（因为是变更）、优先级为“High”。

在“Description”文本框中输入对该需求变更的描述，如图 2-36 所示。

在提交需求变更之后，需求会显示在“Request List”区域中，对应需求变更的具体说明文档。

图 2-36 需求变更提交


2. 计划需求

项目负责人 Lisa_Lu 登录系统，可以看到组内人员的变动情况，显示在“My Teams”区域。由于需求申请之后需要批准，所以刚才新增的需求显示在“Pending Requests”区域中，这里可以拒绝或接受需求申请，如图 2-37 所示。

图 2-37 待批准的需求



质量全面管控：从项目管理到容灾测试

单击  图标, 可以为需求创建任务, 在打开的页面中输入任务名称“Implement Contacts Information can be synchronized”, 再输入对该需求变更任务的描述。


在“Resource(s) Allocation”区域中选择需求开始的时间、预计花费的时间并分配资源, 如图 2-38 所示。将资源分配设置为“Anyone”后, 该项目组中的其他成员就可以登录并获取任务。

图 2-38 需求任务关联

3. 获取需求任务

项目组成员 camus 登录系统, 在“My Tasks”中看到这个任务, 在如图 2-39 所示的确认对话框中接受任务后, 在页面中单击  图标就可以分配到这个任务, 此任务相应的图标也会变为 , 即编辑状态。

图 2-39 确认对话框

获取到任务后, camus 开始执行这个任务。当有状态更新时, 单击  图标, 可以对状



态和实际工时进行更新，如图 2-40 所示。

Plandora (Albert Camus) - Defaults - Out

Task Tracking Form

Project: CRM Development Project Category: Implementation

Name: Implement Contacts Information can be synchronized

Description: Implement contacts information can be synchronized

Hours Task: [dropdown]

Duration: [dropdown]

Estimated Initial Date: 05/10/2016 Actual Initial Date: 05/10/2016

Subs: 0 min

Actual Time: [table with dates and times]

Day TOTAL: [table with dates and times]

Comment: The task was rescheduled.

Relationship: [table with relationship details]

Artifacts: [table with artifact details]

Save Changes: [button] Show Report: [button] Create Ad hoc Task: [button] Main Form: [button]

Tasks of Resource

ID	Name	Project	Reloader	Actual Date	Actual Time	Category	✓	U	U
212	Implement Contacts Information can be synchronized	CRM Development Project	No	05/10	0h	Implementation	✓	U	U
215	Project Enhancement to allow provide update back	CRM Development Project	No	05/10	0h	Maintenance	✓	U	U
222	Implement Contacts Information can be synchronized	CRM Development Project	No	05/10	0h	Implementation	✓	U	U

Refresh: [button] Don't display finished tasks more than one week ago: [checkbox]

图 2-40 需求任务更新

4. 关闭需求任务

当开发人员 camus 完成需求开发后，可以填写实际花费工时，同时将任务状态修改为“Closed”，此任务完成，如图 2-41 所示。

Plandora (Albert Camus) - Defaults - Out

Task Tracking Form

Project: CRM Development Project Category: Implementation

Name: Implement Contacts Information can be synchronized

Description: Implement contacts information can be synchronized

Hours Task: [dropdown]

Duration: [dropdown]

Estimated Initial Date: 05/10/2016 Actual Initial Date: 05/10/2016

Subs: 0 min

Actual Time: [table with dates and times]

Day TOTAL: [table with dates and times]

Comment: The task was rescheduled.

Relationship: [table with relationship details]

Artifacts: [table with artifact details]

Save Changes: [button] Show Report: [button] Create Ad hoc Task: [button] Main Form: [button]

Tasks of Resource

ID	Name	Project	Reloader	Actual Date	Actual Time	Category	✓	U	U
212	Project Enhancement to allow provide update back	CRM Development Project	No	05/10	0h	Maintenance	✓	U	U
222	Implement Contacts Information can be synchronized	CRM Development Project	No	05/10	0h	Implementation	✓	U	U

Refresh: [button] Don't display finished tasks more than one week ago: [checkbox]

图 2-41 需求任务关闭

关闭的任务在“My Tasks”区域中最终状态显示为绿色，如图 2-42 所示。



质量全面管控：从项目管理到容灾测试

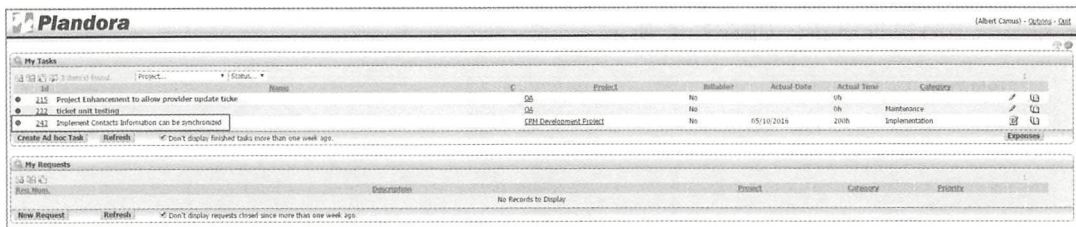



图 2-42 任务关闭状态

5. 查看需求轨迹

项目的需求申请、分配、处理和关闭状态都可以跟踪查看。产品经理 noam 登录系统，在所提交的需求条目中，单击“Details”图标，可以查看跟踪人员、状态和更新时间，如图 2-43 所示。

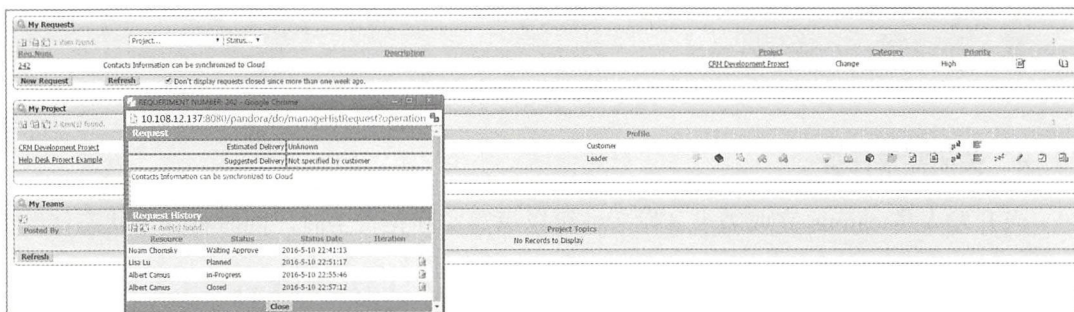


图 2-43 查看需求跟踪轨迹

2.9 要点回顾

- 需求强调的是产品是什么样的，而非产品是怎样设计构造的。
- 需求的收集最终要提交需求说明书来确定产品的设计构造需求。
- 质量保证需求是被定义成可客观性验证的，而不是主观性的。
- 需求误区要防止误入。
- Plandora 管理员配置好相关设置后，前端实际用户就可以开始使用了。
- 前端用户可以管理需求、任务、敏捷板和风险。



第 3 章

代码质量控制

随着互联网产品的推陈出新，客户对软件产品的质量要求也越来越高，希望能够快速对开发工程师编写的代码进行质量检查，并最终给出合理的解决方案，提高软件开发过程中的质量。许多 IT 公司提出自己的代码规范，通过开发团队的内部成员进行手工代码审查来保证软件的产品质量，但是这种方式会浪费大量的人力和财力，不能有效满足当今互联网产品快速迭代的需求，成为了当下必须要解决的一个问题。

由于人工检查代码比较低效，于是出现了静态代码分析工具。它无需运行被测试的代码，通过检查和分析源代码的语法、注释和执行过程来检查代码的规范性，找出代码中隐藏的 errors 和缺陷，例如输入参数不匹配、非法计算、有歧义的嵌套语句及空指针等。开发人员通过执行静态代码扫描分析，减少逐行检查代码的时间，并能快速定位代码中隐藏的 errors 和缺陷，能更专注于分析和解决代码设计的问题，极大地提高了软件的可靠性并节省软件开发的时间和成本。本章将通过如下几个方面介绍自动化代码审查。

- 静态代码分析。
- 代码文档规范。
- Sonar 介绍。
- Sonar 文档规范。
- Sonar 框架搭建。
- Sonar 客户端使用。
- 通过 Jenkins 实现静态代码扫描。



3.1 静态代码分析

传统的代码质量控制一般是阶段性审计代码，在提交代码前，开发人员之间相互审查。这种方式有可能导致代码大量地重改、项目的延期和返工。更糟糕的是，若代码质量差，则造成审查代码花费的时间较长；如果代码数量庞大，那么传统的代码质量控制几乎是无能为力的。

由于传统代码质量控制的局限性，出现了静态代码分析工具 Sonar、PMD、Findbugs 和 Checkstyle 等。它们主要基于缺陷模式匹配、类型推测、模型检查、数据流分析。缺陷模式匹配主要是从代码分析经验中收集足够多的共性缺陷模式，将要分析的代码和已有的共性缺陷模式进行匹配，完成代码分析；类型推测首先定义一套类型机制，包括类型等价、类型保护等规则，然后基于这些规则进行推理计算来检查类型错误，非常适合代码缺陷的快速检查；模型检查主要基于有限状态自动机，首先将要分析的每条语句产生的影响抽象为有限状态自动机的一个状态，通过分析有限状态机而达到分析代码的目的；数据流分析通过收集代码中引用的变量信息，从而分析变量在代码中的赋值、应用及传递等情况。对数据流分析可以确定代码中变量引用的情况，主要检查代码数据流异常，例如先引用后赋值、只赋值无引用等。

由于软件产品的不停迭代，每一次迭代都需要对代码进行审查，在这种情况下出现了持续代码质量检查。它强调整个开发周期的质量，保证内部代码质量问题得到快速反馈和解决。通过持续集成并结合代码质量审查工具，可以提高开发项目的投资回报率，关键是可以尽早发现问题，降低代码修复成本。开发人员提交代码后，进行版本构建，然后自动发送扫描分析报告，一旦发现问题，则会发送短信及邮件通知相关人员，节约发现问题的人力和成本，使问题得到最快的修复。

3.2 代码文档规范

良好的代码规范为后期的代码维护提供了基础，代码的可读性也成为了代码质量的一个关键性指标。良好的代码规范方便开发人员阅读别人编写的代码，不会产生厌恶；可以



通过代码注释了解相关逻辑、输入参数和返回结果，并能顺利找到编写这段代码的时间和联系人等。代码文档规范便于统一风格、传承和交流。公司员工必须使用统一的编辑器，一致的编码风格可以保证代码的可读性。

开发过程中常用的代码规范如下。

- 命名规范：对代码的命名，包含对类、接口、方法和变量名进行规范。
- 编码规范：对代码的格式、长度和编码进行规范。
- 注释规范：对代码注释进行规范。
- 异常处理规范：对代码中异常处理部分进行规范。
- 单元测试规范：对单元测试代码进行规范。
- 文件解析规范：如果代码中有文件操作，则对文件解析进行规范。

以上这些代码规范为开发人员之间的交流提供了桥梁，方便项目的开发和维护，同时实现以下三个目标：

- 规范产品开发过程中采用的技术标准；
- 统一技术，约束同质技术泛滥；
- 对所采用的技术给出适当的使用指南。

3.2.1 命名规范

代码命名规范主要规范文件中的包、类、接口、方法及变量的命名，下面提供的这些规范不需要死记硬背，许多 IDE 工具代码的自动格式化就是遵照以下命名规范的。

- 包：包的名称由小写字母组成，并且以一个顶级域名开头，例如 Java 中包的名称是以 com.开头的。
- 类：类名应该是一个名词，由大写字母和小写字母混合组成，每个单词的首字母大写；使用完整单词，避免缩写词（除非该缩写词被更广泛使用），保持类名简洁而富于描述；建议抽象类以 Abstract 开头，实现类以 Impl 结尾，禁止汉语拼音形式的命名方式。
- 接口：字母大小写的规则与类名相似，不建议以大写字母 I 或 IF 开头。
- 方法：方法名是一个动词，采用字母大小写混合的方式，禁止使用汉语拼音形式的命名方式。第一个单词的首字母小写，其后单词的首字母大写。
- 变量名：第一个单词的首字母小写，其后单词的首字母大写。



- 常量：全部字符由大写字母组成，单词间用下划线隔开。

3.2.2 编码规范

编码规范主要规范代码中的编码、缩进、长度、对齐方式、行宽和间隔。下面列出了代码中的编码规范，供读者参考。

(1) 编码。

- 定义所有代码的编码格式规范，将 Java、JavaScript、HTML、XML、JSP 和各种模板等字符编码方式统一为 UTF-8。
- 重载方法必须使用 `@Override`，避免父类方法改变时导致重载函数失效。
- 不需要警告信息时，可在方法名前使用 `@SuppressWarnings` (“unused”)、`@SuppressWarnings` (“unchecked”) 和 `@SuppressWarnings` (“serial”) 注释。

(2) 缩进。

- 遵循 Java 语言标准的缩进规范，每个层级的缩进量为 4 个空格字符。
- 禁止直接用 Tab 键来缩进，可以将 Tab 键转换为 4 个空格字符。

(3) 长度。

- 为了便于阅读和理解，尽量避免使用大类和长方法。
- 单个方法的有效代码长度应尽量控制在 100 行以内（不包括注释行）。
- 单个方法过长往往造成阅读困难，可以将子方法等相应功能抽取出来，有利于代码的重用度。
- 单个类也不宜过大，当出现此类情况时，可将相应功能的代码重构到其他类中，通过组合等方法调用，建议单个类的长度包括注释行不超过 1500 行。

(4) 对齐方式。

- 关系密切的行应对齐，包括类型、修饰、名称、参数等各部分对齐。
- 如果需要连续赋值，则应对齐操作符。

(5) 行宽。

- 代码行宽设置为 120 字符为宜，超过这个宽度就可能要移动 IDE 的滚动条，造成查看代码不方便。
- 在任何情况下，超长的语句应该在一个逗号或一个操作符后进行换行。

(6) 间隔。



- 类、方法及功能块间应以空行相隔，以增加可读性，但不得有大片空行。
- 操作符两端应当各空一个字符以增加可读性。
- 相对独立的功能模块之间可使用注释行间隔，并标明相应内容。

3.2.3 注释规范

过多的注释会使代码难以阅读，撰写准确且合适的注释是保证高质量代码的关键。Sun Microsystems 公司制定了一套 Javadoc 规范。Javadoc 规范是一个官方注释生成系统，通过注释结构和 Javadoc 标识来自动生成 Javadoc 文档，帮助开发人员阅读代码。

Javadoc 标识由“@”及其后所跟的标记类型和专用注释引用组成。表 3-1 列出了常用的 Javadoc 通用标识，此外还有@serial、@serialField、@serialData、{@docRoot}、{@code}、{@value arg}、{@linkplain}、{@literal}等不常用的标签，这里不展开叙述，读者可以查看帮助文档了解这些标识。通过这些 Javadoc 标识描述代码的功能，生成 Javadoc 文档方便其他人阅读代码。

表 3-1 Javadoc 常用标识

标识符号	用途	应用对象	版本要求
@author	描述开发该类模块的作者信息； @author 可以多次使用，以指明多个作者，生成的文档中每个作者之间使用逗号（,）隔开	类、接口、枚举对象	1.0
@version	描述该类模块的版本信息； @version 可以使用多次，但只有第一次有效	类、接口、枚举对象	1.0
@since	描述何时开始有这个 API 功能模块	类、接口、枚举对象、域、方法	1.1
@see	查看相关内容，如类、方法、变量等	类、接口、枚举对象、域、方法	1.0
@param	描述方法的参数	方法	1.0
@return	描述方法的返回值	方法	1.0
@exception @throws	描述一个抛出的异常	方法	1.2
@deprecated	描述一个已经过时的方法	方法	1.0



续表

标识符号	用途	应用对象	版本要求
{@inheritDoc}	描述一个方法的实现接口	继承方法	1.4
{@link}	链接到其他特定成员对应的文档中	类、接口、枚举对象、域、方法	1.2
{@Override}	重载父类或接口的方法	类	1.5
{@value}	表明静态域的返回值	静态域	1.4

对类、方法、变量注释需要符合 Javadoc 规范，对每个类、方法都应该详细说明其功能、条件、参数等，并使用 HTML 标记格式化注释，以使生成的 Javadoc 文档易于阅读和理解。下面是对 Javadoc 规范的详细说明。

- 一个 Javadoc 块注释以 “/**” 开始，以 “*/” 结束。开始行和结束行之间的注释行都以 “*” 开头。单行注释或者行内的注释以 “//” 作为开始符，注释中的第一个句子要以（英文）句号、问号或者感叹号结束。Javadoc 生成工具会将注释中的第一个句子放在方法汇总表和索引中。
- 为了在 Javadoc 和 IDE 中能快速链接到相关联的类与方法，可尽量多地使用 @see xxx.MyClass、@see xx.MyClass#find(String) 等 Javadoc 标识。
- 标识（java keyword, class/method/field/argument 名, Constants）在注释中第一次出现时以 {@linkxxx.Myclass} 注解，以便 Javadoc 与 IDE 链接。

代码注释的内容也不能太多，太多的注释显得代码烦琐，一个好的注释内容应该包含如下几个方面。

- 注释中的每一个单词都要有其不可缺少的意义，注释里不要写 “@param name - 名字” 这样的废话；如果该注释是废话，连同标签删掉它，而不是自动生成一堆空的标签，如空的 @param name、空的 @return。
- 对于调用复杂的 API，尽量提供代码示例。
- 对于已知的 Bug 需要声明。
- 如果方法允许 Null 作为参数或者允许返回值为 Null，则必须在 Javadoc 中说明，否则方法的调用者不允许使用 Null 作为参数，并认为返回值是 Null Safe（不会返回 Null）。
- 在每个文件、包的头部都应该包含该文件的功能描述、作者、版权、创建和修改记录等。



- 在对类和接口定义之前，应当对它们进行注释，注释的范围包括类和接口的目的、作用、功能、继承于何种父类、实现的接口、实现的算法、使用方法及示例程序。
- 依据标准 Javadoc 规范对方法进行注释，以明确该方法的功能、作用、各参数含义及返回值等。
- 重要的变量加以注释，说明其含义。
- 不易理解的分支条件表达式可加注释。不易理解的循环应说明出口条件。
- 过长的方法实现，应将其语句按实现的功能分段并加以概括性说明。
- 异常处理应注明正常情况及异常情况或者条件，并说明当异常发生时程序应如何处理。
- 参数注释时应注明其取值范围等。
- 返回值应注释失败、错误、异常时的返回情况。
- 异常应当注释什么情况、什么时候、什么条件下会引发什么样的异常。

在 Java 开发中编写出可读性好的代码，需要采用 Javadoc 规范来编写代码注释。采用 Javadoc 规范编写的注释代码示例如下：

```
/**
 * 设置任务模式
 * @author Test
 * @version 1.0.0
 * @date 2016-03-03
 */
public class ArrayClassModeFourAction {
    private int n;
    private int daysOfmonth;
    private int shift[];
    private int year=2015;
    private int month=6;
    /**
     * 员工根据配置模式返回任务列表
     * use {@link #method()} 获取当前日期获
     * @param emplist 员工列表
     * @param year 年
     * @param month 月
     * @param day 日
     * @param ClassMode 配置模式
```



```
    * @return 任务列表
    */
    public List<Schedule> getArrayger(List<Emp> emplist, int year, int month,int day,int ClassMode){
    ...
    }

    /**
    * 获取当前日期
    * @see java.date.RoundingMode
    */
    private void method()
    {
    ...
    }

    /**
    * 执行查询
    *
    * 该方法调用 Statement 的 executeQuery(sql)方法并返回 ResultSet 结果集
    *
    * @param sql 标准的 SQL 语句
    * @return ResultSet 结果集, 若查询失败, 则返回 Null
    * @throws SQLException 查询数据库时可能引发此异常
    */
    public ResultSet executeQuery(String sql) throws SQLException
    {
    ...
    }
}
```

Javadoc 注释完成后, 使用 Apache 组织提供的 Ant 和 Maven 工具, 自动生成 Javadoc 文档, 该文档采用 HTML 方式展示给用户, 通过该文档开发人员能快速了解函数的功能、输入参数及返回的结果类型。

(1) 如果项目使用 Ant 构建, 则在项目的根目录下执行以下命令即可:

```
$ ant javadoc
Buildfile: build.xml
```



Javadoc:

```
[javadoc] Generating Javadoc
[javadoc] Javadoc execution
[javadoc] Loading source file /home/packt/...
[javadoc] Constructing Javadoc information...
```

(2) 如果使用 Maven 构建,则需要安装 Javadoc 插件,编辑 Pom.xml 文件,在<project>节点中添加以下内容,然后执行 mvn site:site 命令。

```
<project>
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>2.9.1</version>
        <configuration>
          <encoding>UTF-8</encoding>
          <charset>UTF-8</charset>
          <docencoding>UTF-8</docencoding>
        </configuration>
        <executions>
          <execution>
            <id>attach-javadocs</id>
            <goals>
              <goal>jar</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </reporting>
  ....
  ....
</project>
```

(3) 通过 Javadoc 工具生成的 Javadoc 文档如图 3-1 所示,生成的 Javadoc 文档保存在该项目的 Target 目录中。





图 3-1 使用 Javadoc 工具生成的文档

3.2.4 异常处理规范

开发人员在编写代码时，经常需要捕捉代码异常，并对捕捉到的异常进行处理，避免干扰程序的正常运行。在程序运行发生错误时，要通过捕捉的异常信息来定位错误来源。在处理代码异常时应注意以下 3 个问题：

- 不进行任何操作，不处理异常，这将导致错误无法跟踪定位；
- 只使用简单的异常信息打印（PrintStackTrace），这将使程序无法对异常进行正确的处理；
- 忽略原来的异常，直接抛出新的异常，这将导致异常堆栈（Exception Stack Trace）断裂，原始的异常信息丢失。

处理异常的正确方式是：首先进行日志记录，需要按规定的格式记录到日志中，包含重要的参数名/值，以方便统计和查找问题；然后根据实际业务逻辑，创建一个测试脚本，并运行这个测试脚本。在运行过程中可能出现异常情况，正确的处理方式是抛出异常信息，打印出异常信息，方便查找问题。

下面的代码用于启动 QTP。

```
Public void RunTestScript()  
{
```



```
try
{
    //创建 QTP 实例
    Log.WriteLine("创建 QTP 实例");
    Application qtApp = new QuickTest.Application();
    qtApp.Launch();

    //创建脚本
    Log.WriteLine("创建脚本");
    qtApp.New(true);
    qtApp.Test.Actions[1].SetScript(strScript);

    //设置主控脚本运行参数
    qtApp.Test.Settings.Run.OnError = "NextIteration";

    //运行测试脚本
    Log.WriteLine("运行主控脚本");
    qtApp.Test.Run(qtResult);
    Log.WriteLine("主控脚本运行结束");
}
catch (Exception ex)
{
    Log.WriteLine("QTP 运行期间异常信息: " + ex.Message);
}
}
```

3.2.5 单元测试规范

有效的单元测试能保证软件产品的质量。下面是对单元测试中代码的规范和建议:

- 所有的 Service、Util、DAO 都必须经过单元测试;
- 测试脚本可以重复执行;
- 测试框架可以使用 JUnit、TestNG、EasyMock 单元测试工具,能有效地保证代码的质量。

3.2.6 文件解析规范

在开发软件产品时,经常要对文件进行操作,例如打开文件、解析文件、下载和上传文件等。如果对文件处理不当,可能会出现内存不能释放、文件忘记关闭、解析文件过大

质量全面管控：从项目管理到容灾测试

造成解析出错等，这些问题都是致命的。下面列出文件解析规范，供读者参考。

1. TXT 文件解析规范

- TXT 文件超过 10 000 条记录则不予解析。
- TXT 文件每行必须有行结束符号\n。
- TXT 文件如果一行有多列数据，则列与列之间的分隔符号要规范。
- TXT 读取文件的流一定要关闭。

实例代码如下（所依赖的包为 java.io，实现用 I/O 流输入输出，根据自定义分隔符解析行数据）：

```
BufferedReader d =
new BufferedReader(new InputStreamReader("FileInputStream"));
String str = null;
while ((str = d.readLine())!=null) {
    coding...
}
```

2. CSV 文件解析规范

- CSV 文件要求不超过 10 000 条记录，大于记录数则不予解析。
- CSV 读取文件的流一定要关闭。

实例代码如下（需要 java.io 和第三方开源包 openCsv.jar，目前 openCsv.jar 使用的版本是 2.2）：

```
BufferedReader d =
new BufferedReader(new InputStreamReader("FileInputStream"));
CSVReader reader = new CSVReaderd);
String[] headers = reader.readNext();
String[] nextLine;
while ((nextLine = reader.readNext()) != null) {
    coding...
}
```

3. PDF 文件解析规范

- 将一个 PDF 文件转换输出为一个文本，通过 Java 操作读取文本，截取所需内容。
- PDF 文件要求不超过 10 000 条记录，大于记录数则不予解析。
- PDF 文件中的图片不解析，只解析文本内容。

- PDF 读取文件的流一定要关闭。

实例代码如下（需要 java.io 和第三方开源包 pdfbox.jar，pdfbox.jar 目前使用的版本是 1.2.1）：

```
PDDocument document = null;
PDFParser parser = new PDFParser("FileInputStream");
parser.parse();
document = parser.getPDDocument();
String text = pdfStripper.getText(document);
Coding...(根据业务需求解析 String)
```

4. Excel 文件解析规范

- Excel 文件要求不超过 10 000 条记录，大于记录数则不予解析。
- Excel 读取文件的流一定要关闭。

代码实例如下（需要 java.io 和第三方开源包 poi.jar，poi.jar 目前使用的版本是 3.6）：

```
POIFSFileSystem fs = new POIFSFileSystem("FileInputStream");
HSSFWorkbook wb = new HSSFWorkbook(fs);
HSSFSheet sheet = wb.getSheetAt(wb.getSelectedTab());
int firstRowNum = sheet.getFirstRowNum();
int lastRowNum = sheet.getLastRowNum();
HSSFRow header = sheet.getRow(firstRowNum);
short firstCellNum = header.getFirstCellNum();
short lastCellNum = header.getLastCellNum();
for(int i=firstRowNum+1;i<=lastRowNum;i++) {
    HSSFRow row = sheet.getRow(i);
    for(short j=firstCellNum;j<=lastCellNum;j++) {
        HSSFCell cell = row.getCell(j);
        Coding...
    }
}
```

5. XML 文件解析规范

- 在使用 Dom4j 时，避免大文件一次读取造成内存溢出，导致文件解析出错停止运行。
- 读取文件的流一定要关闭。

质量全面管控：从项目管理到容灾测试

- 通过 XML 结构定义文档 (XSD) 来验证 XML 的文档格式是否合法，避免处理过程中发现 XML 文档格式不正确而导致资源浪费。

提示：Dom4j 是一个开源 XML 解析包，用于 XML、XPath 和 XSLT 的解析，采用 Java 集合框架，完全支持 DOM、SAX 和 JAXP。

dom4j 开源包网址：<http://sourceforge.net/projects/dom4j>。

SAX 解析优势：它是基于事件的解析方式。分析能够立即开始，而不是等待所有的数据被加载后再处理。由于应用程序只在读取数据时才检查所需数据，因此不需要将数据存储在内存中，这对于大型文档来说是一个突出的优点，因为应用程序不必解析整个文档。

实例代码如下：

```
SAXReader saxReader=new SAXReader();
saxReader.addHandler("/Document/Events/Event/Relation",new XXXDocumentHandler());
saxReader.read("FileInputStream");
private class XXXDocumentHandler implements ElementHandler {
    public void onStart(ElementPath path){
        coding...
    }
    public void onEnd(ElementPath path){
        coding...
    }
}
```

3.3 Sonar 简介

代码质量审查需要相应的工具，常见的审查工具有 Sonar、PMD、FindBugs、CheckStyle 和 Jtest 等，本节介绍目前流行的代码质量分析工具 Sonar。Sonar 是一个代码质量管理的开源平台，与众多的代码质量管理工具相比，它更具有特色和竞争力，其优势主要体现在以下几个方面。

- Sonar 是一个开源的代码质量管理系统，支持包括 Java、C#、C/C++、JavaScript、Groovy 等二十几种编程语言的代码质量管理与检测。
- 通过使用插件机制与 Eclipse、Jira 和 Mantis 等其他外部工具集成，实现对代码质

量的全面自动化分析和管理的。

- Sonar 可以集成不同的代码检测工具，如 PMD、CheckStyle、FindBugs 等。
- 通过使用不同的插件将扫描结果进行再加工处理，采用量化的方式来度量代码质量变化，从而方便地对不同项目类型进行管理。
- 通过与 Jenkins 集成，在代码编译前对代码进行质量检测，通过邮件发送检测结果，尽早检测代码质量，实现持续代码审查。
- 通过 Sonar 代码审查，对代码质量进行如下七个维度的检测。

(1) 复杂度分布：如果文件、类、方法等代码的复杂度过高，会使开发人员难以理解。

(2) 重复性检测：如果程序中包含大量复制、粘贴的代码而导致代码臃肿，则 Sonar 可以展示源码中重复严重的地方。

(3) 单元测试统计：通过 Sonar 可以很方便地统计并展示单元测试覆盖率，开发或测试人员可以了解测试代码的覆盖情况。

(4) 代码规则检测：Sonar 可以通过 PMD、CheckStyle、FindBugs 等代码质量检测工具检测代码是否符合规范。

(5) 注释率统计：若没有代码注释，特别是当不可避免地出现人员变动时，程序的可读性将大幅下降；而过多的注释又会使开发人员将精力过多地花费在阅读注释上。通过 Sonar 的代码注释率统计可以了解代码的注释情况。

(6) 潜在的 Bug：Sonar 可以通过 PMD、CheckStyle、FindBugs 等代码质量检测工具检测出代码中潜在的 Bug。

(7) 糟糕的设计：通过 Sonar 可以找出循环，展示包与包、类与类之间的相互依赖关系，并检测程序之间的耦合度。

3.3.1 Sonar 体系架构

Sonar 是一款代码扫描工具，采用 B/S 架构，通过客户端插件分析源代码。Sonar 客户端可以采用 IDE 插件、Sonar-Runner 插件、Ant 插件和 Maven 插件方式，并通过各种不同的分析机制对项目源代码进行分析和扫描，并把分析扫描后的结果上传到 Sonar 的数据库，通过 Sonar Web 界面对分析结果进行管理。Sonar Web 界面将分析的结果以可视化、可度量的方式展示给用户，用户登录 Sonar Web 界面，就能看到代码分析后的结果和解决方案，如图 3-2 所示。

质量全面管控：从项目管理到容灾测试

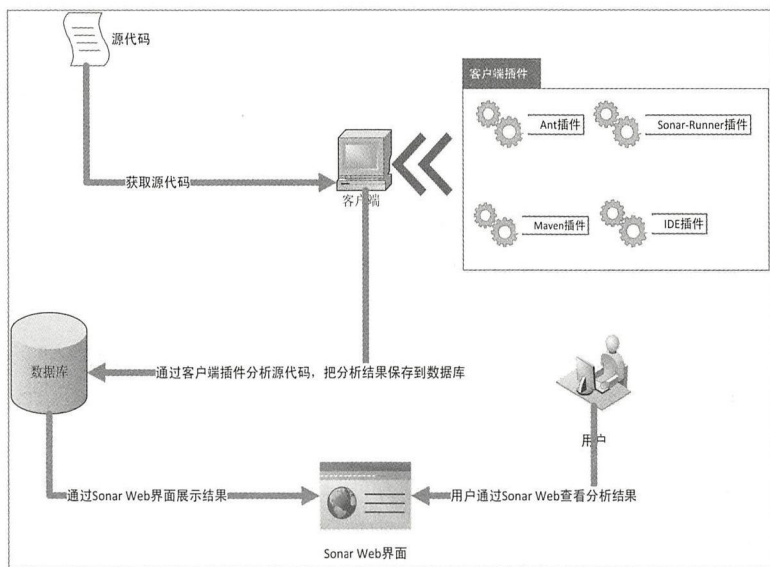


图 3-2 Sonar 体系架构

3.3.2 Sonar 代码规则

Sonar 提供了一套标准来衡量项目的文档和注释，定义了有关注释及文档的 10 条规则，针对前文 Javadoc 和注释结构设定的有关规则，主要可以分为以下两类。

- Javadoc 规则

Sonar 工具根据 Javadoc 规则对 War 包和源代码进行扫描，并扫描代码注释是否符合规则。Sonar 这些规则通过不同的插件（如 PMD、CheckStyle）来定义，并对代码中的包（Package）、方法（Method）和变量（Variable）进行规范，通过这些定义好的规则来对代码扫描。如表 3-2 所示为 Sonar 中常用的 Javadoc 规则。

表 3-2 Sonar 中 Javadoc 规则

严重程度	名称	插件
严重	Undocumented API	PMD
严重	Javadoc Method	CheckStyle
严重	Javadoc Package	CheckStyle
严重	Javadoc Style	CheckStyle
严重	Javadoc Type	CheckStyle
严重	Javadoc Variable	CheckStyle

- 代码内注释规则

如表 3-3 所示为检查空的和未被注释的构造器或方法的规则。

表 3-3 代码内注释规则

严重程度	名称	插件
严重	Uncommented Empty Constructor	PMD
严重	Uncommented Empty Method	PMD
严重	Uncommented Main	CheckStyle
严重	Comment pattern matcher	CheckStyle

在检查文件规则之前，应讨论并细化这些指标。可以从 Sonar 项目的仪表盘中查看注释率和重复率，如图 3-3 所示。



图 3-3 注释率和重复率

这里的 16.6%是指 Sonar 计算出的注释行密度，意味着注释的行数约为总行数的 16.6%。

下面的公式用于计算注释行密度：

$$\text{DCL} = \text{注释行} / (\text{代码行} + \text{注释行}) \times 100\%$$

在 Sonar 中统计代码及注释的标准如下。

- 物理行数：回车的个数。
- 注释行数：注释的行数。
- 有注释的代码行数：拥有注释代码的行数。
- 代码行数：除了空白行、注释、有注释的代码和用于版权的文件头注释以外真正纯代码的行数。
- 公开未注释的 API：没有 Javadoc 注释的 API 个数。
- 公开有注释的 API 密度：有注释公开 API 的个数相对于所有 API 的比例。
- 语句数量：符合 Java 语言规范的语句个数。

质量全面管控：从项目管理到容灾测试

3.4 Sonar 服务端

Sonar 在软件开发过程中变得越来越重要，通过 Sonar 的自动代码扫描能及时发现代码问题，及早解决代码缺陷。通过下面的步骤可以搭建一套基于 Sonar 的代码扫描平台。

3.4.1 环境要求

Sonar 采用 B/S 架构，通过 Web 页面进行交互，搭建 Sonar 平台需要相应的硬件和软件的支持，相关要求如表 3-4 和表 3-5 所示。

表 3-4 安装 Sonar 服务器硬件要求

内容	要求
内存	至少 512MB 内存
CPU	2.50 GHz 以上
磁盘大小	根据项目情况，占用空间大小不同
显卡	4MB 以上

表 3-5 安装 Sonar 服务器软件要求

内容	要求
操作系统	Windows、Linux、Mac、Solaris
JDK	Oracle JDK1.5 以上
数据库	DB2、MySQL、SQLServer
应用服务器	Tomcat5.5 版本以上
浏览器	IE7 以上版本、Firefox、Chrome

3.4.2 Sonar 服务器搭建

(1) 安装 JDK，配置如下 Java 环境变量。

- 新增变量 JAVA_HOME：C:\Java\JDK1.7.0_21（实际安装目录）。
- 添加 Path：%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin。
- 添加 CLASSPATH：%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar。

配置完成后，在命令行中输入“java -version”查看结果，如图 3-4 所示，表示 JDK 安装配置成功。

```
C:\Users\Administrator>java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b01)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

图 3-4 JDK 安装成功信息

(2) 配置 MySQL。

安装完 MySQL 后，新建 Sonar 数据库，赋予数据库访问权限，执行如下 SQL 脚本。

```
CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8_general_ci;
CREATE USER 'sonar' IDENTIFIED BY 'sonar';
GRANT ALL ON sonar.* TO 'sonar'@'%' IDENTIFIED BY 123456;
FLUSH PRIVILEGES;
```

(3) 安装 Sonar。

可以从 Sonar 官网即 <http://www.sonarqube.org/> 下载服务端 Sonar-5.1.2.rar，并解压至 C:\sonar。

(4) 配置 Sonar。

修改 Conf/sonar.properties 文件，配置数据库和端口，配置代码如下：

```
sonar.jdbc.url=jdbc:mysql://127.0.0.1:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance
sonar.jdbc.driver=com.mysql.jdbc.Driver
sonar.jdbc.user=sonar
sonar.jdbc.password=123456
sonar.web.host=127.0.0.1
sonar.web.port=9000
sonar.web.context=/
```

其中各配置项说明如下：

- sonar.jdbc.url 配置 MySQL 地址和编码；
- sonar.jdbc.driver 配置数据库驱动；
- sonar.jdbc.user 配置 Sonar 数据库用户名；

质量全面管控：从项目管理到容灾测试

- sonar.jdbc.password 配置 Sonar 数据库密码；
- sonar.web.host 配置访问的 IP 地址；
- sonar.web.port 配置访问 Sonar Web 界面端口；
- sonar.web.context 配置 Sonar 访问路径。如果 sonar.web.context 为 “/”，则表示访问路径为空，输入 “http://127.0.0.1:9000” 就能访问 Sonar Web 界面。

(5) 设置环境变量，Sonar_HOME=C:\sonar 为 Sonar 解压的路径，在环境变量的路径中添加 “%Sonar_HOME%\bin”，如图 3-5 所示。

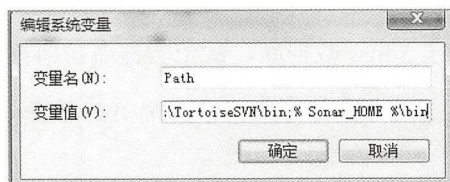


图 3-5 设置 Sonar 环境变量

(6) 在命令窗口中输入 “sonar start”，启动 Sonar 服务端。在浏览器中输入 “http://127.0.0.1:9000” (127.0.0.1 替换为实际 IP 地址)，显示结果如图 3-6 所示。

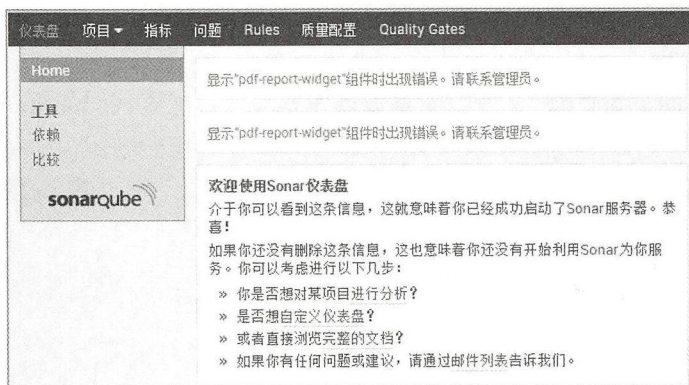


图 3-6 Sonar 主界面

3.4.3 Sonar 配置

(1) 在浏览器中输入 “http://127.0.0.1:9000”，登录 Sonar Web 端后，输入默认用户名 “admin” 和密码 “admin123”，如图 3-7 所示，然后单击 “登录” 按钮。

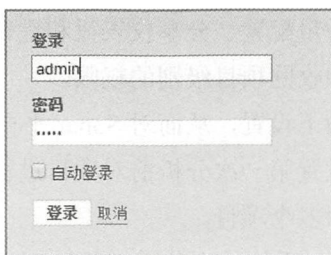


图 3-7 Sonar Web 端登录界面

(2) 进入管理员界面后会显示一个“配置”按钮，单击“配置”按钮可进入 Sonar 配置页面，如图 3-8 所示。

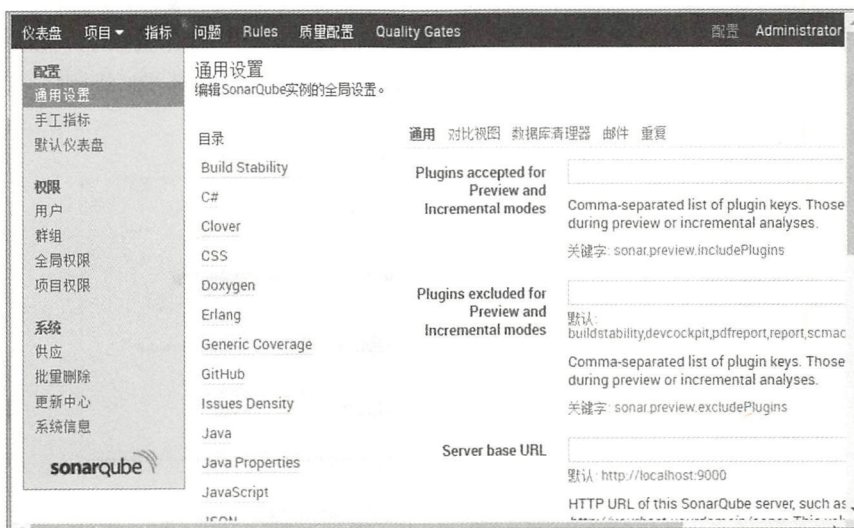


图 3-8 Sonar 配置页面

(3) 在配置模块中可以对 Sonar 进行以下全局设置。

- 通用设置：设置数据清理规则，进行邮件服务器设置。
- 手工指标：可以通过配置接口在项目层次进行指定。
- 默认仪表盘：提供给匿名用户或没有自定义仪表盘的那些用户使用。

(4) 在权限模块中可以对 Sonar 用户和权限进行设置。

- 用户：可增加、修改、删除 Sonar 用户。
- 群组：可创建和管理用户的群组。

- 全局权限：包含修改质量配置、分享仪表盘和进行全局系统管理。
 - 项目权限：设置授予和收回项目级别的权限。
- (5) 在系统模块可以进行如下设置，从而对 Sonar 平台进行更新和维护。
- 供应：初始化项目，设置第一次分析前对项目进行的配置。
 - 批量删除：可一次删除多个项目。
 - 更新中心：主要是 Sonar 系统的安装、卸载和删除插件。
 - 系统信息：描述 Sonar 运行的服务器，包含插件版本、环境变量和数据库。

3.4.4 Sonar 插件

Sonar 提供了大量插件，读者首先要了解被扫描的系统使用的开发语言，然后选择该语言支持的插件进行代码扫描。如图 3-9 所示为 Sonar 支持的语言种类。扫描完成后，可以通过 Sonar 的报表插件显示结果。

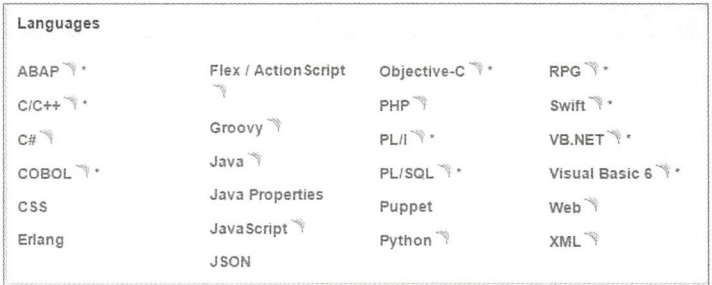


图 3-9 Sonar 支持的语言种类

Sonar 插件的安装可以通过如下两种方式。

- 在 Sonar 官网上手动下载 Jar 包，放到 Sonar 服务器的 lib 目录下，完成插件的安装。
- 由于 Sonar 版本比较多，版本之间的插件可能不一样，因此笔者建议登录 Sonar Web 界面自动安装，通过管理员账号登录 Sonar，在 Sonar 配置菜单中选择系统模块，单击右侧“更新中心”中的选项来安装插件，如图 3-10 所示。

下面介绍一些常用的 Sonar 插件，包含语言插件和 PDF 报告插件，供有兴趣的读者参考和学习。



图 3-10 Sonar 更新中心

● 语言插件

Sonar 默认显示的语言是英文，可以安装 Chinese Pack 插件使 Sonar 支持中文，方便不熟悉英文的读者使用。

● PDF 报告插件

Klicap 是一款可生成 PDF 报告的 Sonar 插件，虽然也有开源版本，但功能有限。它可以产生由 Sonar 接口提供的相关信息，还可以用邮件发送报告，自定义面板，示例如图 3-11 所示。

Klicap 插件的下载网址为 <http://klicap.es/sonarpdfplugin>。

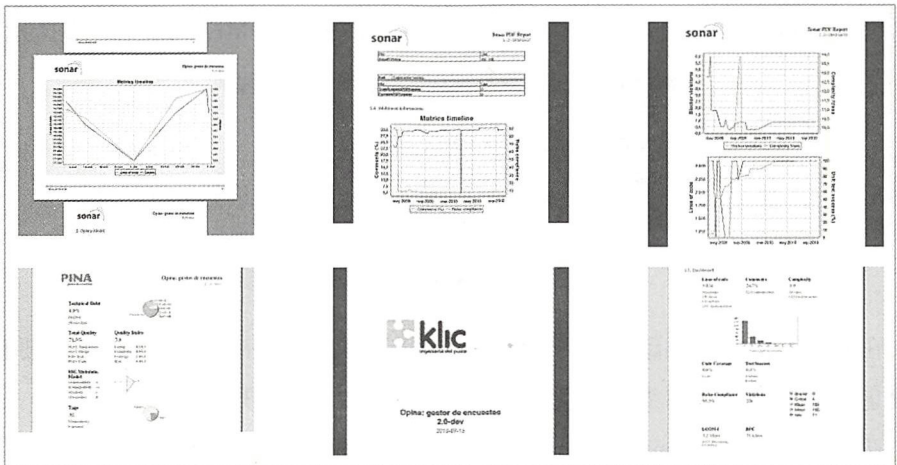


图 3-11 Sonar PDF 报告插件示例

3.5 Sonar 客户端

Sonar 支持多种客户端插件配置，通过不同的方式对代码进行扫描，扫描的结果可以通过 PDF 进行展现。Sonar 扫描的方式主要有以下四种：

- Sonar-Runner 客户端；
- Maven 插件方式；
- Ant 插件方式；
- Eclipse 插件方式。

3.5.1 Sonar-Runner 客户端

(1) 下载 Sonar-Runner 到客户端并安装，修改 `conf/sonar-runner.properties` 配置文件，配置要访问的 Sonar 服务和 MySQL 服务器地址。

```
#---Sonar 服务器地址
sonar.host.url=http://127.0.0.1:9000/
sonar.jdbc.url=jdbc:mysql://127.0.0.1:3306/sonar?
useUnicode=true&characterEncoding=utf8
```

(2) 设置 sonar-runner 的环境变量 `SONAR_RUNNER_HOME`，如图 3-12 所示。

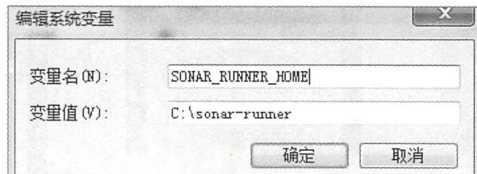


图 3-12 设置 Sonar-Runner 环境变量

(3) 在系统变量中添加路径 “`%SONAR_RUNNER_HOME%\bin`”，注意不要删除原来的路径变量，否则会造成原来的系统命令不能使用，如图 3-13 所示。

(4) 在命令行运行 “`sonar-runner -h`”，结果如图 3-14 所示，表示安装成功。

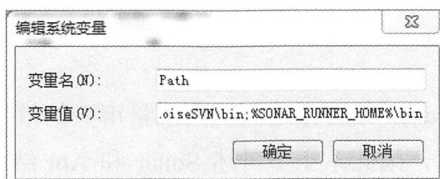


图 3-13 设置 Sonar-Runner 路径变量

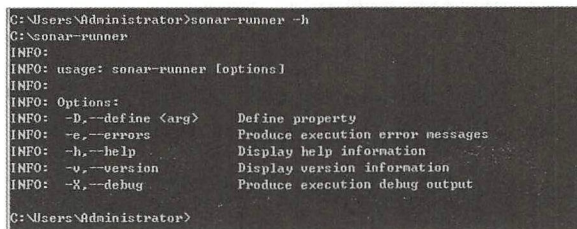


图 3-14 Sonar-Runner 安装成功界面

3.5.2 Maven 插件方式

Maven 集成了 Sonar 的插件，如果读者采用 Maven 方式来管理项目，则只需在 Maven 的配置文件 pom.xml 中添加如下信息，然后运行 mvn sonar:sonar 即可完成 Sonar 代码扫描任务。

```
<profiles>
  <profile>
    <id>sonar</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <sonar.jdbc.url>
jdbc:mysql://127.0.0.1:3306/sonar?useUnicode=true&characterEncoding=utf8
      </sonar.jdbc.url>
      <sonar.jdbc.driver>com.mysql.jdbc.Driver</sonar.jdbc.driver>
      <sonar.jdbc.username>sonar</sonar.jdbc.username>
      <sonar.jdbc.password>123456</sonar.jdbc.password>
      <sonar.host.url>http://127.0.0.1:9000</sonar.host.url>
    </properties>
  </profile>
</profiles>
```


3.5.3 Ant 插件方式

前面讲述了 Sonar 和 Maven 结合来达到代码质量审查的效果，但是 Maven 的学习成本高，并不是任何项目都适合，因此本小节讲述 Sonar 和 Ant 结合来进行代码质量审查。就目前来说，Sonar 和 Ant 的集成没有做到 Sonar 和 Maven 的集成那么完善，因此在使用过程中需要多写一些脚本。下面介绍操作步骤。

(1) 使用 Ant 插件，首先要下载 sonar-ant-task-2.2.jar 包到 test/lib 目录中，如图 3-15 所示。

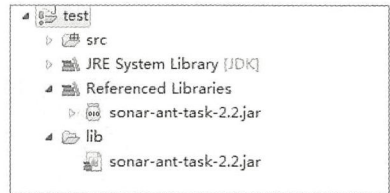


图 3-15 Ant 中的 Sonar 插件

(2) Ant 使用 build.xml 文件来定义任务，通过已定义的任务完成代码的扫描，编辑源码根目录中的 build.xml 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test " default="all" basedir="." xmlns:sonar="antlib:org.sonar.ant">
<!-- =====定义项目属性 ===== -->
  <property name="src.dir" value="src" />
  <property name="test.dir" value="test" />
  <property name="lib.junit.dir" value="lib" />
  <property name="build.dir" value="target" />
  <property name="classes.dir" value="{build.dir}/classes" />
  <property name="reports.dir" value="{build.dir}/reports" />
  <property name="reports.dir" value="{reports.dir}/junit" />
  <!-- SonarQube properties 配置 -->
  <property name="sonar.projectKey" value="Sonar Test " />
  <property name="sonar.projectName" value="Sonar Test " />
  <property name="sonar.projectVersion" value="1.0" />
  <property name="sonar.language" value="java" />
  <property name="sonar.sources" value="{src.dir}" />
  <property name="sonar.tests" value="{test.dir}" />
  <property name="sonar.binaries" value="{classes.dir}" />
  <property name="sonar.sourceEncoding" value="UTF-8" />
  <property name="sonar.surefire.reportsPath" value="{reports.dir}" />
  <!-- Sonar 数据库配置 -->
  <property name="sonar.jdbc.url" value="jdbc:mysql://127.0.0.1:3306/sonar?useUnicode=true&
characterEncoding=utf8&rewriteBatchedStatements=true" />
  <property name="sonar.jdbc.username" value="sonar" />
  <property name="sonar.jdbc.password" value="123456" />
```



```
<!--定义 target 清理数据-->
<target name="clean">
    <delete dir=".sonar" />
    <delete dir="${build.dir}" />
    <delete dir="${reports.dir}" />
</target>
....
<!-- 定义 Sonar Target-->
<target name="sonar" depends="compile">
    <taskdef uri="antlib.org.sonar.ant" resource="org/sonar/ant/antlib.xml">
        <classpath path="lib/sonar-ant-task-*.jar" />
    </taskdef>
    <!-- 执行 Sonar -->
    <sonar:sonar />
</target>
</project>
```

(3) 进入源码的根目录并执行命令“ant”，完成后访问 <http://localhost:9000/>，即可看到测试结果，如图 3-16 所示。

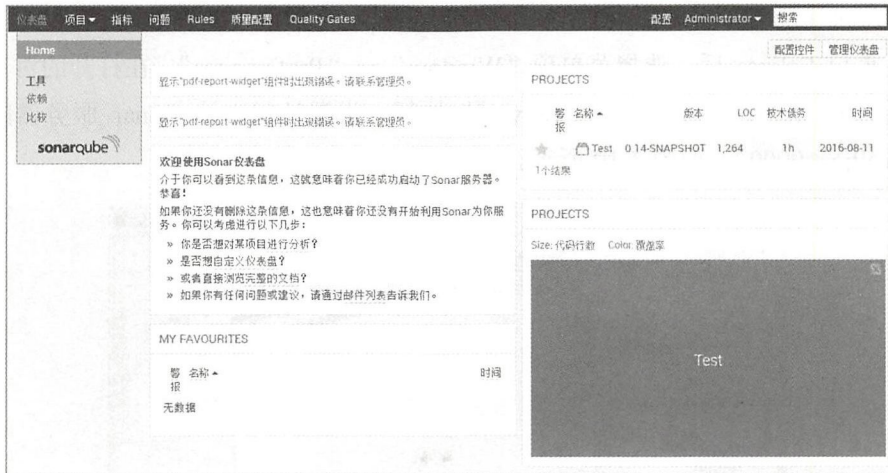


图 3-16 Sonar 测试结果

3.5.4 Eclipse 插件方式

Java 编辑器 Eclipse IDE 提供 SonarLint 插件对代码进行扫描，通过这个插件可以方便地



扫描自己的代码，尽早发现代码中的问题。下面通过 Eclipse 的 IDE 来配置 SonarLint 插件。

(1) 打开 Eclipse，新建一个 Java 项目，项目的名称设置为“Sonarlint-demo”。

(2) 选择菜单项“Help”→“Eclipse Marketplace”，如图 3-17 所示，进入 Eclipse 插件安装界面。

(3) 打开 Eclipse Marketplace 窗口，在“Find”文本框中输入“SonarLint”，单击“Install”按钮开始安装 SonarLint 插件，如图 3-18 所示。

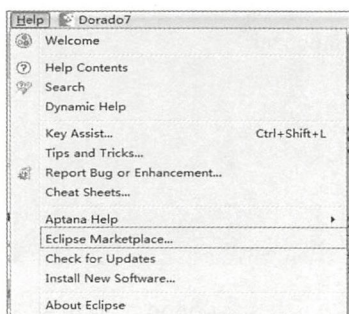


图 3-17 选择“Eclipse Marketplace”

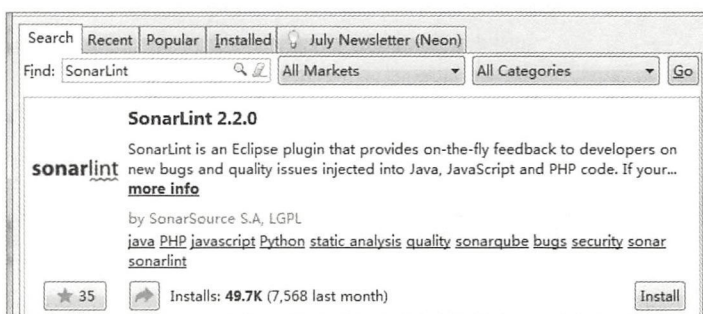


图 3-18 安装 SonarLint 插件

(4) SonarLint 插件安装完成后，需要重启 Eclipse，插件才能生效。

(5) 重启 Eclipse 后，选择菜单项“Window”→“Preferences”，在打开的对话框的左侧列表中找到 Sonar 选项，单击“Servers”按钮，设置访问本地 Sonar 服务器的地址为“http://localhost:9000”，如图 3-19 所示。

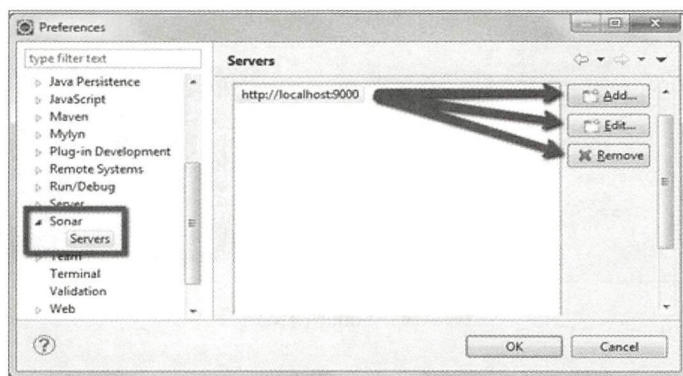


图 3-19 配置 Sonar 服务器

(6) Sonar 服务器配置完成后，下一步是将 Eclipse 项目链接到 Sonar 服务器，并利用



Sonar 服务器进行分析。首先在 Project Explorer 中用鼠标右键单击项目，在打开的快捷菜单中选择“Configure”→“Associate with Sonar”，如图 3-20 所示。

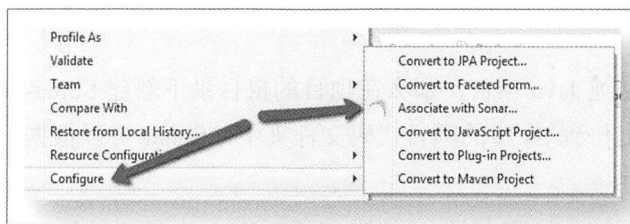


图 3-20 链接项目到 Sonar 服务器

(7) 在“Eclipse Project”文本框中输入项目名称“Sonarlint-demo”，然后在“SonarQube Project”文本框中输入“SonarLint for Eclipse (parent)”进行绑定，如图 3-21 所示。

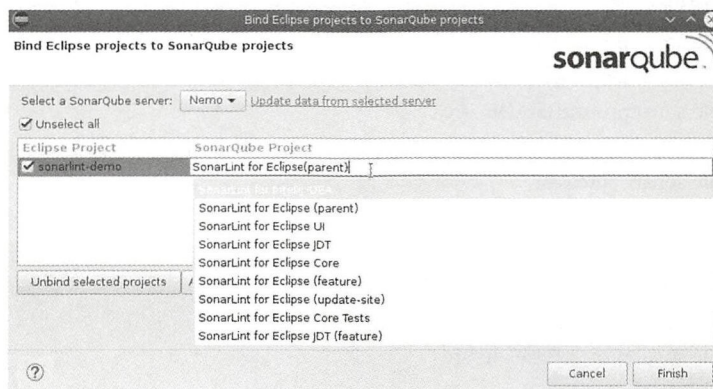


图 3-21 配置项目关联 Sonar

(8) 单击“Finish”按钮后，项目会关联到 Sonar 服务器上并进行分析，登录 Sonar Web 端可以查看扫描结果。

3.6 最佳实践

前面已经讲解了 Sonar 框架的搭建和客户端的使用，可以在项目中方便地使用 Sonar 进行代码扫描。下面通过持续集成来实现静态代码扫描的工作，在使用持续集成前需要搭建 Jenkins 服务器，通过 Jenkins 和 Sonar 完成静态代码扫描。



3.6.1 项目配置

(1) 如果项目的源代码采用 SVN 进行管理，那么新建 Java 项目并上传项目代码到 SVN 服务器。

(2) 如果项目是纯 Java 项目，那么在项目的根目录下新建 Sonar-project.properties，添加扫描配置信息并上传到 SVN 的项目代码文件夹中，Sonar 可以根据配置文件信息进行代码扫描，配置信息如下。

```
sonar.projectKey=App-Web-Sonar
sonar.projectName=App-Web-Sonar
sonar.projectVersion=1.0
sonar.modules=java-module,javascript-module,web-module

#扫描 Java 代码
java-module.sonar.projectName=Java Module
java-module.sonar.language=java
java-module.sonar.projectBaseDir=src
# .表示 projectBaseDir 指定的目录
java-module.sonar.sources=.

#扫描 Web 项目
web-module.sonar.projectName=Web Module
web-module.sonar.language=web
web-module.sonar.projectBaseDir=src
web-module.sonar.sources=.

#扫描 JavaScript
javascript-module.sonar.projectName=JavaScript Module
javascript-module.sonar.language=js
javascript-module.sonar.projectBaseDir=src
javascript-module.sonar.sources=.
```

下面对上面的代码做一些说明。

- sonar.projectKey：项目唯一的标识。
- sonar.projectName：在 Sonar Web 端显示的名称。
- sonar.projectVersion：项目版本信息。
- sonar.modules：扫描时需要包含的模块。



- `xx.sonar.sources`: 源代码目录。
- `xx.sonar.language`: Sonar 扫描时使用的语言。

(3) 如果项目使用 Maven 构建, 那么在 Maven 的 `Pom.xml` 文件中, 需要配置 SonarQube 插件信息, 可以参考 3.5.2 小节 Maven 插件方式, 本小节不再详细介绍。

3.6.2 持续审查

(1) 登录持续集成系统 Jenkins, 新建一个 App-Web-Sonar 项目, 如图 3-22 所示。

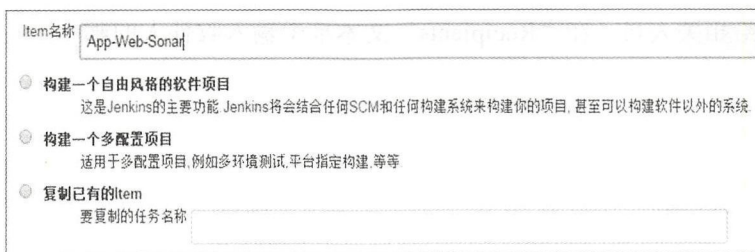


图 3-22 新建 Sonar 项目

(2) 在扫描过程中需要获取项目的源代码, Jenkins 支持多种获取源代码的方式, 包括 Git、SVN、SCM 等, 本小节使用 SVN 管理项目源代码, 在“源码管理”页面中输入需要扫描源代码存放的 SVN 地址, 然后输入用户名和密码, 如图 3-23 所示。

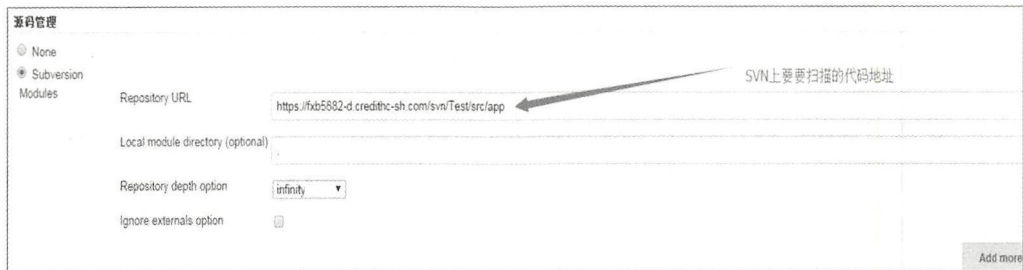


图 3-23 配置 SVN

(3) 配置构建策略, 例如在日程表中设置成每天下午 6:00 到 SVN 上获取代码, 如图 3-24 所示。然后执行代码扫描, 并把结果显示在 Jenkins 上, 其中“H 18 * * *”表示每天下午 6:00 从 SVN 上获取代码, 它的格式类似于 Linux 中的 Crontab。



构建触发器

☐ Build after other projects are built

☒ Build periodically

日程表

H 18 * * *

Would last have run at 2016年9月17日 星期六 下午06时12分17秒 CST. would next run at 2016年9月18日 星期日 下午06时12分17秒 CST.

☐ Poll SCM

图 3-24 配置构建策略

(4) 在“构建后操作”页面中，如图 3-25 所示，选择“E-mail Notification”选项可以自动发送邮件给相关人员，在“Recipients”文本框中输入收件人的邮箱，在部署失败时会发送邮件给收件人，单击“保存”按钮保存配置信息。

构建后操作

☒ E-mail Notification

Recipients

Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

☒ 每次不稳定的构建都发送邮件通知

☐ 单独发送邮件给对构建造成不良影响的责任人

删除

增加构建后操作步骤

保存 应用

图 3-25 配置邮件信息

(5) 返回主界面，选择刚刚创建的 Jenkins 项目，单击“立即构建”按钮，Jenkins 就会自动从 SVN 上获取开发人员提交的源代码，然后进行自动化扫描，并把扫描分析结果存放到 Sonar 服务器。

(6) 构建完成后，扫描结果会显示在 Sonar Web 界面中，在浏览器中输入 Sonar Web 界面的地址“http://127.0.0.1:9000/”即可查看扫描结果，如图 3-26 所示。如果读者想使用邮件方式发送 Sonar 生成的扫描结果，可以配置 Jenkins 的邮件模板，同时 Sonar 还需要配置 3.4.4 小节所介绍的 PDF 报告插件，配置完成后，Jenkins 会把 Sonar 扫描后的 PDF 文件通过邮件发送给相关人员。

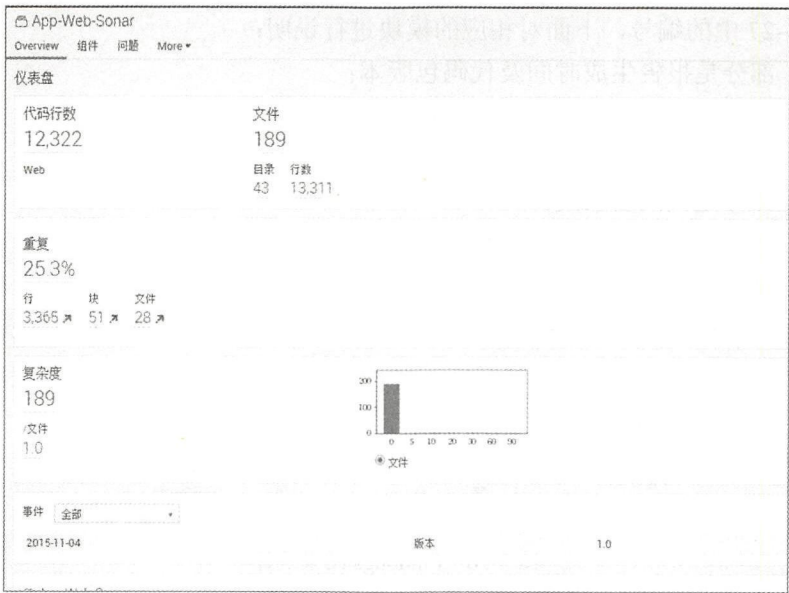


图 3-26 查看运行结果

3.6.3 结果分析

Sonar 报告主要分为 6 个部分，如图 3-27 所示。

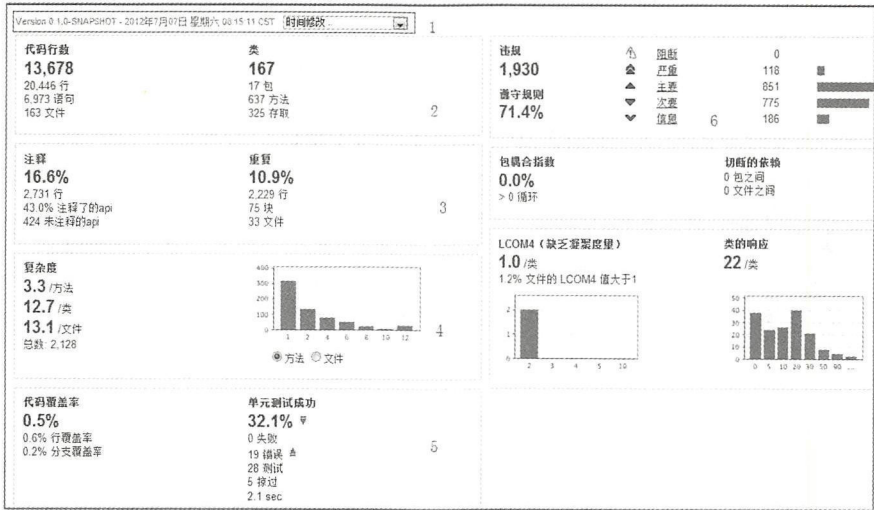


图 3-27 Sonar 报告



根据图 3-27 中的编号，下面对相应的模块进行说明：

- 第 1 部分是报告生成时间及代码包版本；
- 第 2 部分是项目规模（包括代码行、类、包数目的统计）；
- 第 3 部分是项目注释和重复情况；
- 第 4 部分是代码复杂度；
- 第 5 部分是代码覆盖率及单元测试情况；
- 第 6 部分是 Findbugs 结果统计。

Findbugs 主要描述了代码出错的情况，如图 3-28 所示。单击报告中的数据可以进入详细数据页面。

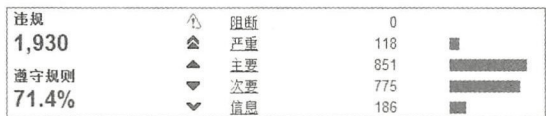


图 3-28 Findbugs 结果统计

Findbugs 结果统计中还列出了错误的等级。其中，阻断为零，严重 118 个，主要 851 个，次要 775 个，信息 186 个。单击 Findbugs 中的数据统计，可以发现如下问题。

- 单击报告中的问题等级链接，如图 3-29 所示，可以查看违规的等级严重性。

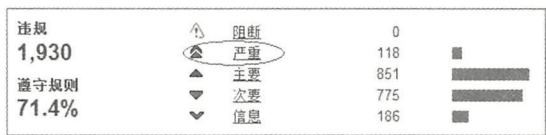


图 3-29 Findbugs 中的问题等级链接

- 单击需要查看的模块，如图 3-30 所示，在页面中可以查看代码中各个模块的详细信息。

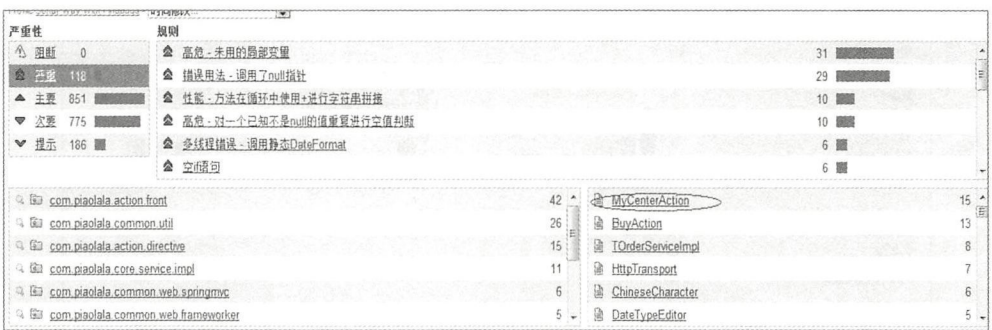


图 3-30 查看代码中各个模块的详细信息

- 单击代码有问题的类 MyCenterAction, 可以查看该类的详细代码, 如图 3-31 所示。其中前面的红色箭头为问题等级标识。

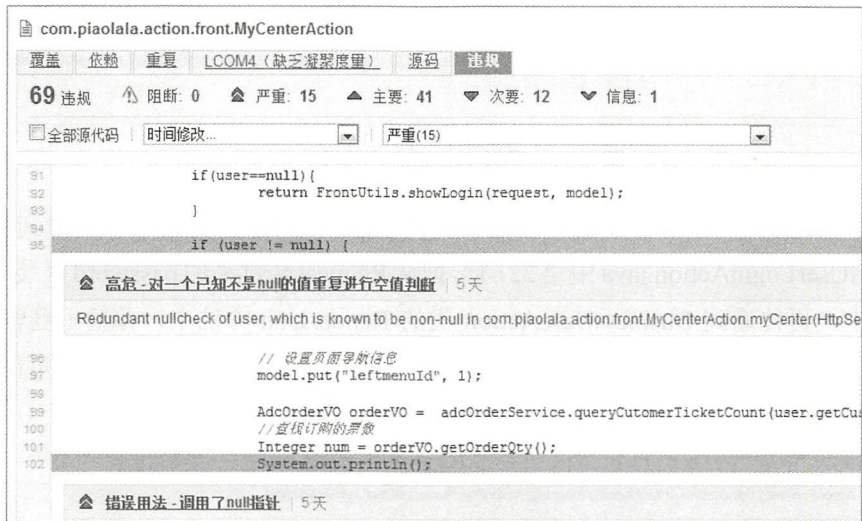


图 3-31 查看违规代码

- 通过 Findbugs 可以查看有错误提示的类名和包名, 如图 3-32 所示, 单击相应类名可以查看具体的错误信息。

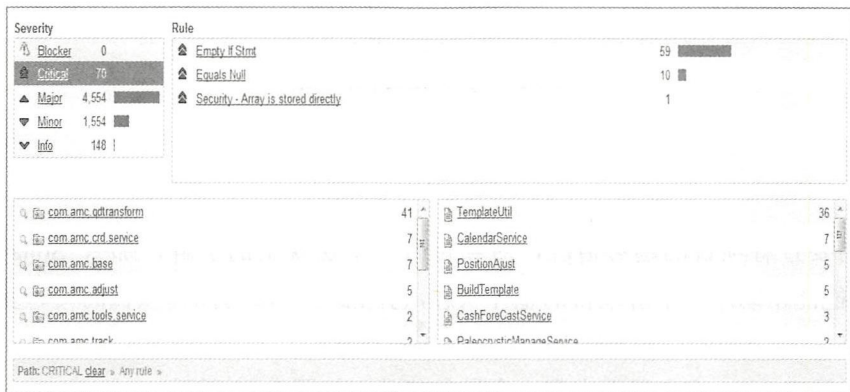


图 3-32 查看错误的类名和包名

- 如图 3-33 所示的代码行 81 和行 84, 显示了“Equals Null”警告, 通过查看这些警告, 开发人员可以及时修改这些代码。

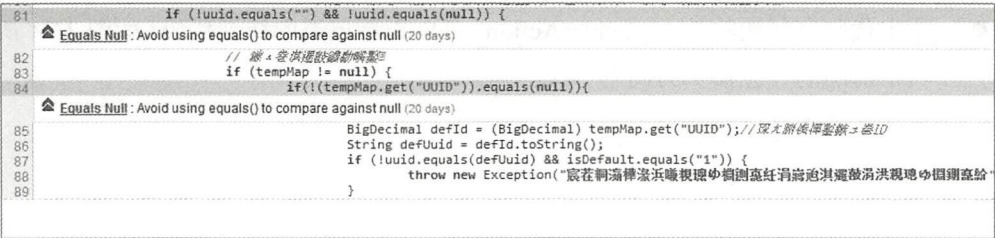


图 3-33 通过 Findbug 查看警告信息

- 如图 3-34 所示为 Sonar 扫描出的传送敏感信息的风险。Sonar 扫描到 Java 文件 GUserLoginAction.java 中第 221 行，把从 Request 传过来的 password 封装于 bossDto 中，再传递到 BusinessBaseAction 类的 RenderJson 方法中，并输出到客户端。在这个过程中并没有对 Password 进行加密，从而导致被黑客嗅探到的可能。建议在传递敏感数据之前进行加密，如使用 SHA-256 加密法。



图 3-34 扫描出的敏感信息

3.6.4 集成曲线图

如果需要查看项目的趋势情况，对项目有一个宏观上的了解，那么 Sonar 的 Time machine 功能能帮助用户了解项目的整体状态，它能显示一段时间内项目的代码复杂度和单元测试覆盖率等。使用 Time machine 的操作步骤如下。

- (1) 登录 Sonar，在主界面中有一个导航菜单，如图 3-35 所示。
- (2) 在导航菜单中单击“Time machine”菜单项，在打开的页面中可以查看该项目质量的趋势情况，包括代码复杂度、单元测试覆盖率和 Findbugs 结果的线性增长情况，如图 3-36 所示。

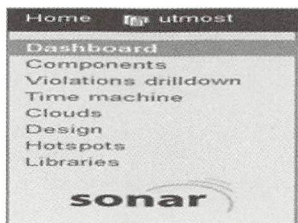


图 3-35 Sonar 主界面

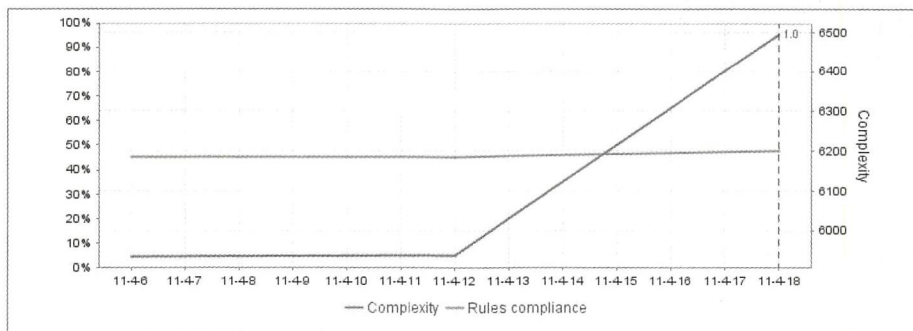


图 3-36 查看项目质量的趋势

3.7 要点回顾

软件质量控制是为了保证软件代码的质量，是软件工程中一个不可缺少的环节。通过本章的学习，读者可了解代码质量管理对提高项目质量意义重大。Sonar 是一个基于插件、开源的代码扫描工具，提供各种方式的客户端，包含 IDE、Maven、Ant 和 Sonar-Runner 等。通过 Sonar 和 Jenkins 集成并构建一套自动代码扫描方案，及时高效地对开发人员的代码进行检查，并把扫描结果通过邮件和 Web 界面展示给开发人员，避免了人工代码审查，为项目的稳定提供了质量保证。

静态代码扫描工具主要基于缺陷模式匹配、类型推测、模型检查和数据流分析。随着代码的复杂性越来越高，静态代码扫描工具可能出现漏报和误报的问题。从表面上看，误报（false Positive）和漏报（false Negative）是一对无法兼容的矛盾体，要想漏报率低就可能误报，要避免误报就可能导致漏报。因此，读者不要完全依赖静态分析工具，需要结合人工审核减少误报和漏报，保证项目的质量。

第 4 章

自动化部署

前面已经介绍了软件代码的质量控制，但是如何把高质量的代码发布到开发环境、测试环境甚至生产环境呢？代码部署时要保证软件能正确地部署到服务器上，供用户使用；在用户使用过程中出现问题时，能及时回退到上一个版本，或者增加一些补丁包来修复错误。如果采用手工部署的方式，则需要手工获取代码、打包和发布，导致部署时间过长，而且由于系统配置的复杂性，不能保证软件部署完成后就能正常使用，于是出现了自动化部署的概念。自动化部署主要解决手工部署流程化的问题，通过一键部署发布到不同的测试环境或者生产环境，减少人工操作的失误，缩短系统发布时间。

自动化部署主要包含代码提交、构建出包、集成部署、集成测试、上线会签、系统部署、生产验证、跟踪监控和上线总结等信息流程管理。自动化部署需要实现高度的自动化，减少人工依赖，降低错误发生的几率，提高端到端流程的执行效率；提供人工干预及审批流程，随时跟踪，及时发现问题，在发现问题后及时进行处理。这就形成了一个完善的部署平台，不仅包含系统部署，还对版本进行管理。如果系统出现问题，能及时地回退到上一个版本；通过配置中心实现部署的标准化，把配置信息集中到配置中心统一管理，如果需要修改配置信息，只需要在配置中心进行修改，不用重新部署系统和重启系统来解决系统的配置问题。如果业务系统出现故障，则通过监控系统报警，并及时通知相关人员修复这些问题，避免造成不必要的损失。自动化部署如此重要，本章从如下几个方面介绍自动化部署。

- 认识自动化部署。
- 配置中心的优化。

- 持续集成的实现。
- 影响自动化部署的因素。
- 版本控制。
- Git 客户端使用。
- 基于 JDeploy 平台搭建自动化部署。

4.1 引入自动化部署

在软件开发过程中，项目成员非常期待使用自动化部署来发布应用程序，因为手工部署容易出错并且容易使人产生厌倦和疲劳感。当一个新的软件产品发布完成后，后期就需要对该软件进行升级和维护。手工部署有哪些缺陷？自动化部署能解决什么问题？自动化部署过程中容易出现哪些问题？读者可以通过下面的内容来了解自动化部署。

4.1.1 复杂的手工部署

对于目前大多数应用程序来说，无论规模大小，其部署过程都比较复杂，而且包含很多非常灵活的部分。许多公司或组织依然使用手工方式来发布软件，也就是说部署应用程序的操作步骤都是由相对独立的个人或某个小组来分别执行。软件的手工部署流程如图 4-1 所示。

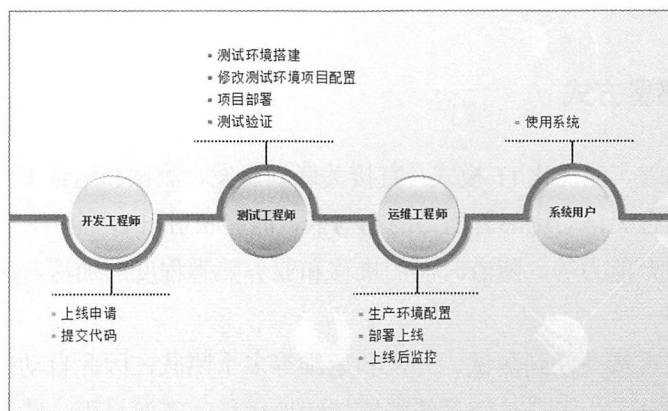


图 4-1 手工部署流程

(1) 上线申请：当软件开发测试完成后，由开发团队的成员填写上线申请单，申请单的内容即软件上线的部署步骤。

(2) 提交代码：申请上线通过后，开发人员需要提交代码到服务器。

(3) 测试环境搭建：测试人员为了保证能够升级部署成功，需要复制生产环境中的程序和数据到本地的测试环境中，搭建类似于生产环境中的测试环境。

(4) 修改测试环境项目配置：根据开发人员提交的代码，修改项目配置参数，如数据库的连接、依赖的服务接口地址。

(5) 项目部署：编译并打包部署项目到应用服务器。

(6) 测试验证：根据上线申请单中所描述的上线部署步骤进行验证。

- 验证不通过：如果发现上线部署步骤有问题，则与开发人员进行沟通，让开发工程师修改上线部署步骤。

- 验证通过：通知运维人员，准备部署生产环境，运维工程师需要配置生产环境。

(7) 生产环境配置：测试环境验证通过后，运维工程师部署生产环境，包含硬件（如CPU、内存、网络）和软件（如操作系统、数据库、应用服务器、中间件）环境。

(8) 部署上线：运维工程师登录到生产环境中，依照上线部署步骤手工操作完成。

(9) 上线后监控：运维工程师对生产环境的应用进行监控，保证项目平稳运行。

(10) 使用系统：通过一系列的发布、测试后，用户开始使用软件系统。

因为每个步骤里都有一些细节需要进行人为判断，所以很容易发生错误。即使不是这样，这些步骤的执行顺序和时机的不同也会导致结果的差异，这种差异很可能带来不良的后果，而自动化部署则可以减小这些影响。

4.1.2 自动化部署方式

引入自动化部署，能优化 IT 模式，直接关联到研发、测试和运维多个团队，可以成为一家公司的核心平台。自动化部署能力的高与低，能够映射出 IT 能力，例如流程、环境管理、服务耦合和平台能力等。随着时间的推移和业务熟悉程度的加深，应使手工部署过渡到完全自动化部署。

由于部署系统直接关联到测试、开发和运维等多个团队，因此自动化系统部署成为公司基础架构的一部分。为了让这个系统能够持续地运行，需要采用一些必要的开发实践和技术。可以使用部署流水线将高度自动化的测试和部署及全面的配置管理结合在一起，实

现一键式软件发布。也就是说，只需要单击一下鼠标，就可以将软件部署到任何目标环境，包括开发环境、测试环境和生产环境。在这个过程中，通常的做法有以下三种：

- 采用自动化脚本（Shell、Python、Ruby）来执行系统部署；
- 自主研发自动化部署平台完成自动化部署；
- 在 Jenkins 中，通过插件来实现自动化部署。

4.1.3 持续集成思想

随着软件项目复杂度的增加，频繁的软件集成产生了持续集成，它是一种软件开发实践，即团队成员不断地集成他们的工作，通过自动化构建（编译、打包、测试）来验证自己集成的模块是否有错误，从而尽可能早地发现错误并且解决错误。

持续集成是全自动化的一个终极体现，它的主要流程是通过版本控制库来控制版本，自动获取源码完成项目的构建，构建完成后进行自动化测试，测试完成后发送测试报告给相关人员，如果出现问题立即通知相关人员进行修复；如果这个流程设计成定时任务，一旦开发人员提交代码到版本控制库中就会自动触发构建、测试、生成报告的流程。持续集成需要各个方面的资源协同工作才能完成任务，并且整个流程自动化，减少了人工的干预。持续集成的主要特点如下。

- 全自动化：一个项目配置好后，若配置要求不变，就不需要修改，根据配置好的流程自动运行即可。
- 开发人员不断地把代码提交到版本控制库中，持续集成会自动下载代码进行构建。
- 构建完成后进行自动化测试，如果测试没有通过，则发送邮件或报告给相关责任人。
- 开发人员可以集中精力编写功能代码，持续集成会不断地测试开发人员编写的代码，出现错误会通知开发人员。
- 持续集成在多人协同工作时更加有效。如果某个开发人员上传的代码没有测试通过，就能及时发现问题，保证项目代码顺利合并。

除了持续集成，还有一个持续部署，它是持续集成在自动化测试通过后，然后自动部署到测试环境或者生产环境的过程。有了持续集成，持续部署使代码合并、测试、发布串成一条线，简化了部署工作，开发人员可将精力主要放在代码的开发和设计中，为高质量的软件奠定了基础。

4.2 自动化部署的特点

读者已经了解了自动化部署的重要性，有了自动化部署后能方便开发测试和更好地管理系统的发布任务，避免人为操作错误。下面从三个方面介绍自动化部署，逐渐形成一套完整的自动化部署流程。

- 环境一致性。
- 部署系统化。
- 配置集中化。

4.2.1 环境一致性

环境管理通常比较复杂。一般来说，环境包括几个层次，即硬件及网络配置、操作系统、应用程序所依赖的中间件、应用程序运行时所需的数据及其配置。目前对于硬件及网络配置、操作系统这两层来说，有两种方式进行管理，一种是利用专用软件进行自动化远程配置，可以通过一些技术对机器进行自动安装和配置；另一种是使用虚拟化技术进行系统集中管理。

在力所能及的情况下，要保持测试环境和生成环境的一致。例如，若生产环境可能是 100 台机器的集群，那么测试环境至少需要两台或四台机器作为集群；若生产环境使用 Tomcat，那么测试环境和开发环境也应该使用相同的 Tomcat，最好保证版本也是一致的。上线部署时尽管环境配置文件的内容不一样，但是操作流程基本相同，所有的机器软件和硬件也相同，所以上线部署时脚本出错的几率就比较小了。而且，这种自动化部署没有人工干预，能够避免手工误操作。

4.2.2 部署系统化

自动化部署需要在一个系统里集中管理编译、打包、测试和发布的流程，开发人员提交代码到 Git 或 SVN 服务器后，自动化系统部署执行。如图 4-2 所示简要描述了自动化系统部署的工作方式和流程。

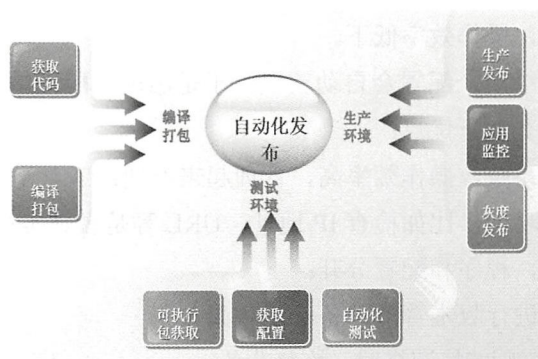


图 4-2 自动化部署的工作方式和流程

(1) 获取代码：开发人员提交代码后，自动化部署平台从 Git 服务器上获取代码，完成代码的合并。

(2) 编译打包：自动化部署平台获取代码后，对代码进行编译和打包。

(3) 可执行包获取：获取编译后的安装包，自动发布到开发测试环境。

(4) 获取配置：从配置中心（Disconf、Salt）获取配置信息。

(5) 自动化测试：应用程序发布到测试环境，执行自动化测试，保证常规业务正常运行。

(6) 应用监控：把代码发布到开发环境后，需要查看发布日志，通过日志监控查看服务器运行情况。

(7) 灰度发布：测试环境测试通过后，把代码发布到生产环境中的其中一台机器上，进行堡垒测试，并观察这台机器的发布日志。

(8) 生产发布：堡垒测试通过后，通过自动化部署平台自动部署到生产环境中的其他机器上，然后监控生产环境中的服务器日志，无异常即完成部署。

4.2.3 配置集中化

在软件发布的过程中，经常会遇到软件的配置问题，不同的环境其配置也不一样。在系统数量不多的情况下，一般使用各自的系统管理各自的配置。如果系统数量很多，则可能会有如下问题：

- 系统上线时，要对多个系统的配置进行检查，沟通协调耗费大量的人力；
- 各个系统自己管理配置，通过文本来编辑配置，没有自动校验机制，配置容易出错；
- 在配置过程中，需要人工来检查，如果配置信息过多，则很难发现错误；

质量全面管控：从项目管理到容灾测试

- 配置信息分散，操作效率低下。

引入自动化发布系统后，配置会自动获取，于是出现了配置中心，它集中管理各个系统的配置。配置中心具有如下优点：

- 集中配置多个系统，操作效率高，管理起来方便；
- 提高配置检查功能，比如检查 IP 地址、URL 等是否真实存在；
- 使配置集中化，程序和配置分开；
- 对系统的配置进行版本管理和备份，如果业务出现问题，则能快速搭建，恢复业务。

目前有许多配置中心，例如 SaltStack、Disconf 和 ZooKeeper 等，它们集中管理各个系统的配置。如图 4-3 所示是 Disconf 的拓扑图，X、Y、Z 业务系统都从 Disconf 获取配置信息，而不需要在各个业务系统中设置配置信息。

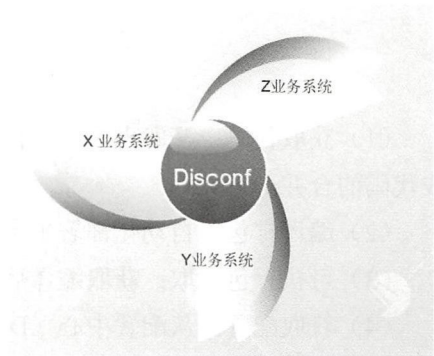


图 4-3 Disconf 拓扑图

Disconf 主界面如图 4-4 所示，其中包含了主要的配置，如 APP、实例列表、操作等各个系统的配置信息，可以对配置信息进行删除、修改和更新操作。修改配置信息后不需要重启系统即可生效，实现配置和系统相分离，保证了配置和系统的独立性。



图 4-4 Disconf 主界面

4.3 版本控制

在自动化部署中离不开版本的控制，如果本次发布出现问题，影响了用户使用，就需要回滚至上一个版本，并且把上一个版本自动发布到生产测试环境，保证用户能正常使用。利用版本控制系统来管理源代码，能追踪代码修改的历史。版本控制是自动化部署的基础设施，自动化系统部署需要通过版本管理软件来获取源代码和版本信息，然后对源代码进行编译和发布等。下面介绍常用的版本控制系统 Git。

4.3.1 Git 简介

随着软件需求的不断更新，版本不停地迭代，这时就需要一个版本控制软件追踪多个版本的开发和维护活动。在编程界有这样一种说法：没有版本控制的项目，就等于没有这个软件。常用的版本控制软件很多，过去使用 CVS、SVN，现在流行 Git。随着开源产品的流行，使用 Git 的用户也越来越多，它的重要性是不可估量的。Git 是 Linux 之父 Linus Torvalds 开发的一个版本控制软件，主要是为了方便管理源代码，促进开发人员之间的合作，加强对版本的管理。随后 Git 越来越流行，逐渐取代了原来的 CVS、SVN 等版本控制系统。Git 具有如下特点：

- Git 是开源、免费的分布式版本控制系统；
- 可从服务器上克隆完整的 Git 仓库（包括代码和版本信息）到单机上；
- 可在自己的机器上根据不同的开发目的创建分支、修改代码；
- 可在单机上自己创建的分支上提交代码；
- 可在单机上合并分支；
- 适合分布式开发，强调个体；
- 任意两个开发者之间可以很容易地解决冲突；
- 速度快、灵活。

Git 的工作方式和 SVN 不同，它使用本地仓库和远程仓库来存储源代码。如图 4-5 所示简要描述了 Git 的工作方式，可以看出 Git 包含四个工作区域：远程仓库、本地仓库、暂存区和工作区，它们中间通过不同的操作来管理相关区域，具体操作步骤如下。

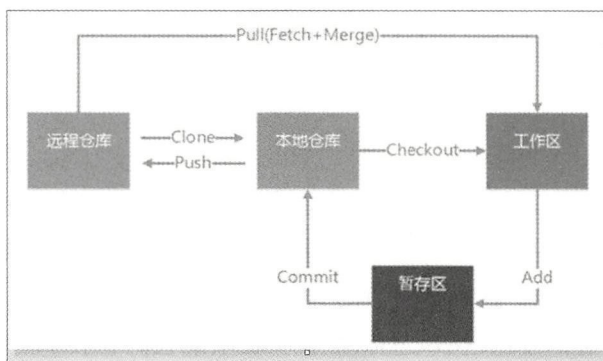


图 4-5 Git 工作流程

(1) 从远程仓库克隆到本地仓库时会新建一个目录，这个目录就是 Git 目录，用来保存元数据和对象数据库。Git 目录非常重要，每次克隆仓库时，实际上复制的就是这个目录里面的数据。从项目中取出某个版本的所有文件和目录称为工作区，用以进行后续工作。

(2) 暂存区只不过是一个简单的文件，有时候人们会把这个文件叫做索引文件，但是术语称之为暂存区。在工作区添加文件后会进入暂存区，它是对修改或者添加文件的一个快照。

(3) 提交后，暂存区的文件快照永久存储到本地仓库的 Git 目录中。

(4) 在本地仓库进行推送，会把本地仓库的文件同步到远程仓库，完成文件的更新添加操作。

(5) 如果本地工作区需要更新或合并远程仓库文件，则需要从远程仓库拉取或者合并所有文件到本地工作区。

4.3.2 Git 部署

Git 是一个开源的版本管理系统，实现代码版本的管理，可通过 Web 界面新建公开或者私人项目。使用 Git 能够浏览源代码，管理缺陷和注释。通过 Git 的权限管理可以管理团队对仓库的访问。可以浏览提交过的版本和文件历史库。Git 还提供一个代码片段收集功能，可以轻松实现代码复用，便于日后有需要的时候进行查找。下面介绍 Git 搭建的方法。





1. 环境要求

Git 采用 B/S 架构，通过 Web 页面进行交互，搭建 Git 服务器需要相应硬件和软件的支持，相关的硬件要求和软件要求如表 4-1 和表 4-2 所示。

表 4-1 Git 硬件要求

项目	要求
内存	至少 512MB 内存
CPU	2.50 GHz 以上
磁盘大小	根据项目情况占用空间大小不同

表 4-2 Git 软件要求

项目	要求
操作系统	Centos 7 以上版本
浏览器	IE7 以上版本、Firefox、Chrome

2. 操作步骤

因为 Git 是使用 Ruby on Rails 开发的，因此可以用 Linux 环境搭建。在 Linux 环境中搭建 Git 的步骤如下。

(1) 配置 Git 服务器的同时需要安装邮件服务，需要开放 HTTP 和 SSH 端口，命令如下：

```
sudo yum install curl openssh-server postfix cronie
sudo service postfix start
sudo chkconfig postfix on
sudo lokkit -s http -s ssh
```

(2) 通过 Curl 下载 Git 安装包并执行安装脚本，命令如下：

```
curl http://packages.gitlab.cc/install/gitlab-ce/script.rpm.sh | sudo bash
```

(3) 使用 rpm 命令安装 Git 安装包，命令如下：

```
rpm -i gitlab-ce-8.2.3.x86_64.rpm
```

(4) Git 安装完成后，在浏览器的地址栏中输入“127.0.0.1”，显示 Git 登录界面，如图 4-6 所示，表示 Git 安装成功。





质量全面管控：从项目管理到容灾测试

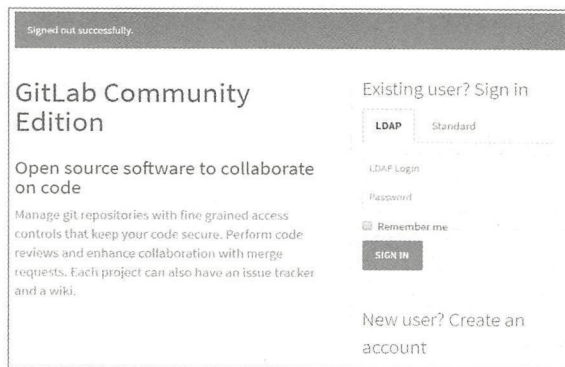


图 4-6 Git 登录界面

3. 配置 Git

Git 服务器安装完成后,需要配置 `gitlab_ssh_host` 地址和 `gitlab_host` 地址,配置步骤如下。

(1) 打开 `gitlab.rb` 文件,对其进行编辑,命令如下:

```
vi /etc/gitlab/gitlab.rb
```

(2) 编辑 `gitlab.rb` 文件,添加 `gitlab_host` 和 `gitlab_ssh_host` 两个选项,其中“127.0.0.1”建议修改为本机 IP 地址。'

```
##external-url-for-gitlab
external_url = 'http://127.0.0.1'
## Note: configuration settings below are optional.
## Uncomment and change the value.
#####
# gitlab.yml configuration #
#####
gitlab_rails['gitlab_host'] = '127.0.0.1'
gitlab_rails['gitlab_ssh_host'] = '127.0.0.1'
```

(3) 由于 Git 实际运行的是 `gitlab.yum` 文件,所以需要将 `/etc/gitlab/gitlab.rb` 配置信息同步至 `/var/opt/gitlab/gitlab-rails/etc/gitlab.yml` 文件中,命令如下:

```
sudo gitlab-ctl reconfigure
```

(4) 配置完成后,需要重启 Git 服务器,让刚才的配置生效,命令如下:

```
sudo gitlab-ctl restart
```





4.3.3 Git 客户端使用

配置好 Git 服务器后, 就可以使用 Git 客户端对 Git 进行操作了。Git 客户端有 SourceTree、GitUp、SmartGit 和 QGit 等, 建议将 MsysGit 和 TortoiseGit 相结合来操作 Git。MsysGit 通过命令行方式操作 Git, 而 TortoiseGit 给用户提供一个友好的界面供用户操作 Git。如果读者不熟悉命令行, 则可以使用 TortoiseGit 工具。下面详细介绍这两款工具的使用方法。

1. MsysGit

(1) 如果是 Windows 系统, 使用 Git 前需要下载 MsysGit 软件, 下载完成后双击安装文件, 其欢迎界面如图 4-7 所示, 单击“Next”按钮开始安装。

(2) 弹出 MsysGit 所遵循的 GNU 协议, 如图 4-8 所示。使用 MsysGit 必须同意 GNU 协议, 然后单击“Next”按钮。



图 4-7 安装 MsysGit 欢迎界面

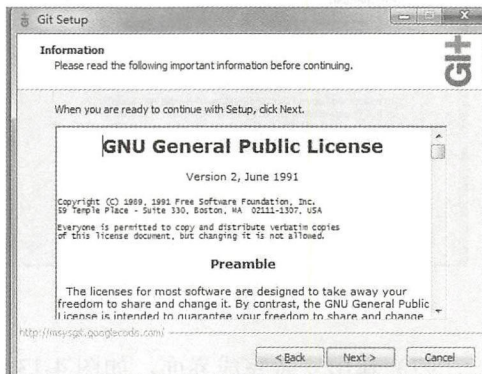


图 4-8 MsysGit 遵循的 GNU 协议

(3) 单击“Browser”按钮, 在打开的对话框中选择 MsysGit 需要安装的位置, 如图 4-9 所示, 然后单击“Next”按钮。

(4) 弹出路径环境变量设置对话框, 如图 4-10 所示。选择默认项“Use Git Bash Only”, 然后单击“Next”按钮。

(5) 弹出选择 SSH 可执行文件对话框, 选择默认项“Use OpenSSH”来执行 SSH 协议, 如图 4-11 所示, 然后单击“Next”按钮。

(6) 在打开的对话框中选择默认的“Checkout Windows-Style, commit Unix-style line endings”选项, 如图 4-12 所示, 然后单击“Next”按钮。



质量全面管控：从项目管理到容灾测试

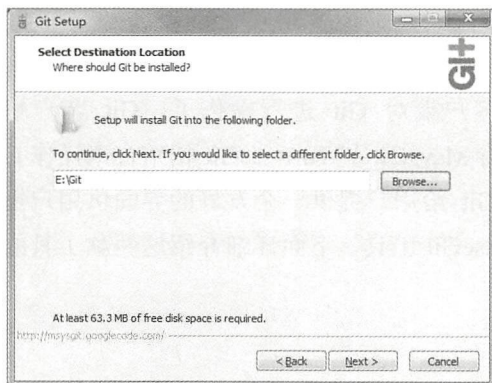


图 4-9 设置 MsysGit 安装目录

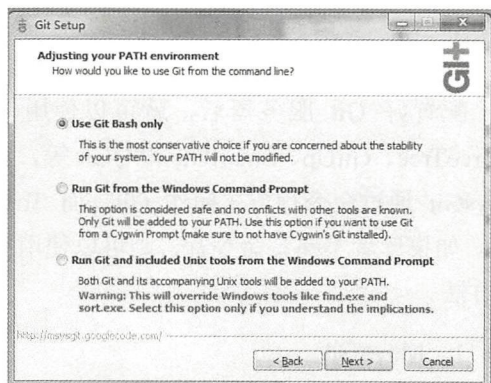


图 4-10 设置路径环境变量

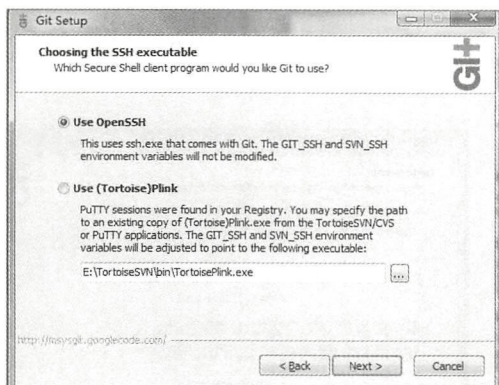


图 4-11 选择 OpenSSH 协议

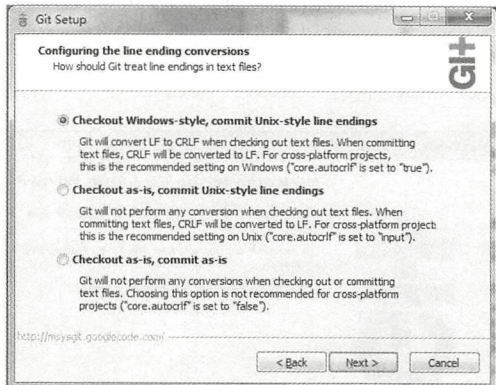


图 4-12 选择 MsysGit 的代码样式

(7) 弹出安装完成界面，如图 4-13 所示，单击“Finish”按钮退出安装界面。



图 4-13 MsysGit 安装完成界面





2. TortoiseGit

TortoiseGit 软件主要为不熟悉命令行的 Git 用户准备的,它提供人性化的界面帮助用户操作 Git,如果用户熟悉 TortoiseSVN,那么 TortoiseGit 的使用方式和 TortoiseSVN 一样,选择文件或者文件夹后单击鼠标右键就会弹出 Git 操作菜单,包含新建版本库、创建分支、拉取代码和提交代码等。TortoiseGit 安装比较方便,下载 TortoiseGit 安装包之后,双击安装包进行安装,一直单击“Next”按钮就可以了。

提示:注意,TortoiseGit 安装路径的名字里面最好不要出现空格,建议新建一个专用文件夹来安装 TortoiseGit。

TortoiseGit 安装完成后,在任何工作目录中单击鼠标右键,弹出的快捷菜单如图 4-14 所示,其中“Git Clone”和“TortoiseGit”选项用于 Git 的操作。

为了更加方便地使用 TortoiseGit 软件,需要通过 MsysGit 对 TortoiseGit 进行配置。



图 4-14 快捷菜单

(1) 运行已经安装好的 MsysGit 软件, MsysGit 界面类似 Windows 的命令行窗口,如果用户名为 Test,则输入如下命令配置 Git 的用户名。

```
git config --global user.name Test
```

(2) 配置好 Git 的用户名后继续配置 Git 的用户邮箱,如果用户的邮箱是 Test@qq.com,则输入下面的命令配置 Git 的用户邮箱。

```
git config --global user.email Test@qq.com
```

(3) 配置结束后,在桌面上单击鼠标右键,在弹出的快捷菜单中选择“TortoiseGit”→“settings”,在打开的“Settings-TortoiseGit”对话框的左侧列表框中选择“Git”,右侧显示 Git 设置选项,“Name”文本框中显示的是刚才设置的用户名 Test,“Email”文本框中显示了刚才设置的用户邮箱 Test@qq.com,表示 TortoiseGit 配置成功,如图 4-15 所示。





质量全面管控：从项目管理到容灾测试

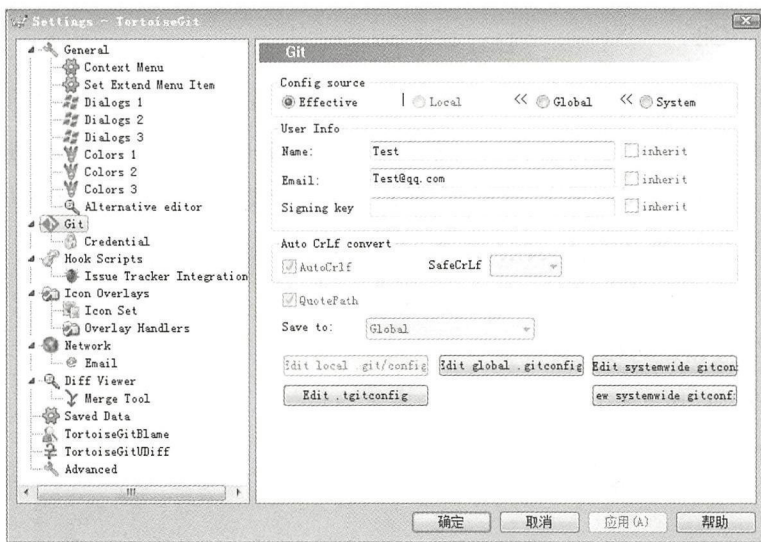


图 4-15 TortoiseGit 配置成功

4.3.4 Git 相关操作

MsysGit 安装完成后，就可以使用 Git 命令了，Git 命令有很多，例如从远程仓库中获取代码、更新代码到远程仓库等，但是常用的 Git 命令也就几十个。下面简要介绍使用 Git 过程中经常使用的命令。

(1) 对 Git 进行初始化配置，包含用户名、邮件、缓存时间等，经常会用到以下命令：

```
#配置使用 Git 仓库的用户名
git config --global user.name "Your Name Comes Here"
#配置使用 Git 用户的 E-mail
git config --global user.email you@yourdomain.example.com
#配置到缓存，默认是 15 分钟
git config --global credential.helper cache
#修改缓存时间
git config --global credential.helper 'cache --timeout=3600'
#列举所有配置
git config -l
```

(2) 取得 Git 仓库，首先初始化一个版本仓库到本地，然后复制远程版本库，再添加远程版本库，查看远程仓库的信息，常用的命令如下：





```
#初始化一个版本仓库
git init
#复制远程版本库
git clone url
#添加远程版本库 origin
git remote add origin url
#查看远程仓库
git remote -v
```

(3) 创建好 Git 仓库后，可以提交需要修改或者添加的代码，在提交代码前需要把文件添加到暂存区，然后提交修改的代码到本地仓库，并推送到远程服务器，可以通过如下命令来提交查看、修改、添加、跟踪及删除文件。

```
#添加当前修改的文件到暂存区
git add .
#提交修改并添加注释
git commit -am "注释"
#推送更新到远程服务器，语法为 git push [远程名] [本地分支]:[远程分支]
git push origin master
#查看文件状态
git status
#添加新文件并跟踪
git add readme.txt
#从当前跟踪列表移除文件并完全删除
git rm readme.txt
#仅在暂存区删除，保留文件在当前目录，不再跟踪
git rm --cached readme.txt
#重命名文件
git mv reademe.txt readme
#查看提交的历史记录
git log
#假设你已经使用 git add .，将修改过的文件 a、b 加到暂存区
#现在只想提交 a 文件，不想提交 b 文件
git reset HEAD b
#取消对文件的修改
git checkout -- readme.txt
#删除一个文件
git rm [file name]
#查看已经被提交的文件
git ls-files
```





(4) 有时候开发人员需要查看自己修改的记录，可以通过如下命令来查看：

```
# 查看该文件每次提交的记录
git log <file>
# 查看每次详细修改内容的 diff
git log -p <file>
# 查看最近两次详细修改内容的 diff
git log -p -2
#查看统计信息
git log --stat
```

(5) Git 可以通过本地来管理远程仓库，例如查看远程仓库的名称、查看仓库的状态、修改和添加远程仓库等。下面是几个常用的管理远程仓库的命令。

```
# 查看远程服务器地址和仓库名称
git remote -v
# 查看远程服务器仓库状态
git remote show origin
# 添加远程仓库地址
git remote add origin git@127.0.0.1:test/test.git
# 设置远程仓库地址(用于修改远程仓库地址)
git remote set-url origin git@127.0.0.1:test /test.git
# 删除远程仓库
git remote rm <repository>
```

如表 4-3 所示为 Git 的常用命令，有兴趣的读者可以找一本 Git 的专业图书来详细研究 Git 的使用方式，本书不再赘述。

表 4-3 Git 常用命令

操作类型	命令
检出仓库	git clone [url]
查看远程仓库	git remote -v
添加远程仓库	git remote add [name] [url]
删除远程仓库	git remote rm [name]
修改远程仓库	git remote set-url --push [name] [newUrl]
拉取远程仓库	git pull [remoteName] [localBranchName]
推送远程仓库	git push [remoteName] [localBranchName]
显示命令的帮助信息	git help <command>





续表

操作类型	命令
显示某次提交的内容	git show
比较两次提交之间的差异	git diff <\$id1> <\$id2>
在两个分支之间比较	git diff <branch1>..<branch2>
查看远程分支	git br -r
创建新的分支	git br <new_branch>
删除某个分支	git br -d <branch>
将 branch 分支合并到当前分支	git merge <branch>
切换到某个分支	git co <branch>
创建新的分支并且切换过去	git co -b <new_branch>
基于 branch 创建新的 new_branch	git co -b <new_branch> <branch>

4.3.5 代码管理

下面介绍使用 TortoiseGit 来管理项目的源代码，对项目进行创建、提交、推送等相关操作的方法。

(1) 登录 Git，新建一个项目，项目名称为 API-Atf-Demo，在“Visibility Level”选项区中选择“Private”，即只允许被授权的用户看到该项目。如果选择“Internal”，则登录 Git 的用户都能看到该项目；如果选择“Public”，则不用登录就能看到该项目。本例选择“Private”，建立一个私有 Git 项目，单击“Create Project”按钮完成创建，如图 4-16 所示。

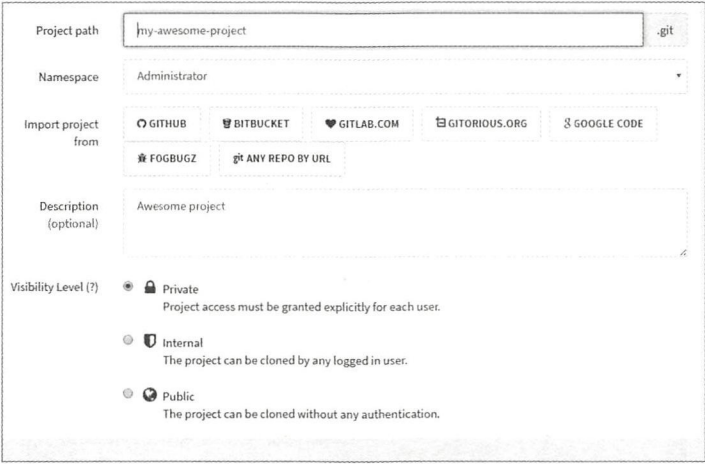


图 4-16 通过 Git 新建项目



(2) 新建项目完成后进入项目详情页，可以查看项目的 URL，如图 4-17 所示。这个 URL 就是项目 API-Atf-Demo 的 Git 地址，后面的代码都会提交到这个地址。



图 4-17 新建项目的 URL

(3) 在本地目录新建一个文件夹，如 API-Atf-Demo。选择这个文件夹，单击鼠标右键，在弹出的快捷菜单中选择“Git Clone”，弹出“Git Clone-TortoiseGit”对话框，如图 4-18 所示。在 URL 文本框中输入上一步提到的“Git URL”地址，在“Directory”中选择刚刚新建的 API-Atf-Demo 文件夹，单击“OK”按钮后，会提示输入用户名和密码，输后会自动获取 API-Atf-Demo 项目的代码。

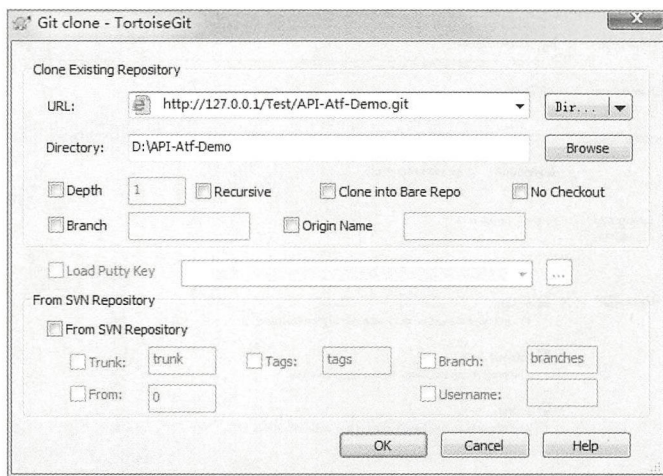


图 4-18 获取项目代码



(4) 获取 API-Atf-Demo 项目的代码后, 在该文件夹的图标上会有一个绿色的标志, 如图 4-19 所示。如果为红色标志, 则表示该文件夹中的内容没有同步到 Git 仓库。出现红色的叉号表示该文件夹中的文件和 Git 仓库中的文件有冲突, 需要解决。

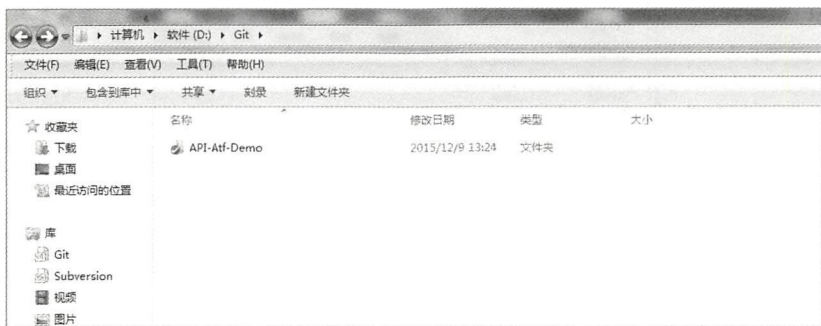


图 4-19 拉取项目代码文件夹

(5) 如果需要添加新文件, 则该文件夹会显示一个蓝色的加号。选择文件夹后单击鼠标右键, 在弹出的快捷菜单中选择 “TortoiseGit” → “Add” 选项, 把该文件添加到暂存区, 然后单击 “Commit” 按钮提交代码到本地仓库, 提交成功之后的提示如图 4-20 所示。

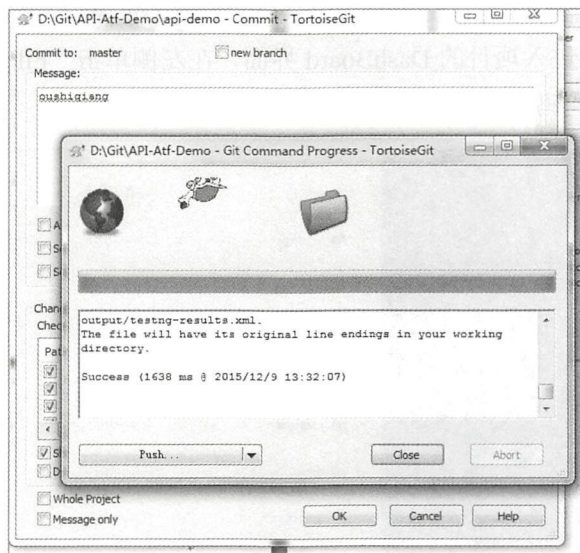


图 4-20 提交代码

(6) 代码提交到本地仓库后, 还未同步到远程仓库, 此时在如图 4-20 所示的对话框中



单击“Push”按钮，弹出如图 4-21 所示的对话框。在该对话框中单击“OK”按钮，将代码提交到远程服务器。

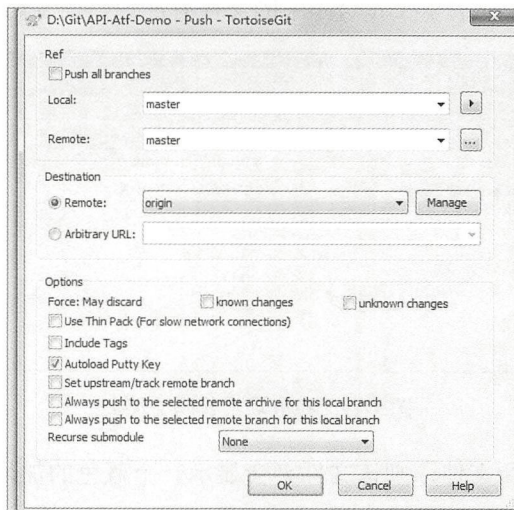


图 4-21 提交项目代码

(7) 如果想查看代码是否提交成功，可以登录 Git Web 界面，选择刚才新建的 API-Atf-Demo 项目，进入项目的 DashBoard 界面，在左侧单击“Files”菜单项就可以看到刚才提交到 Git 上的代码，如图 4-22 所示。

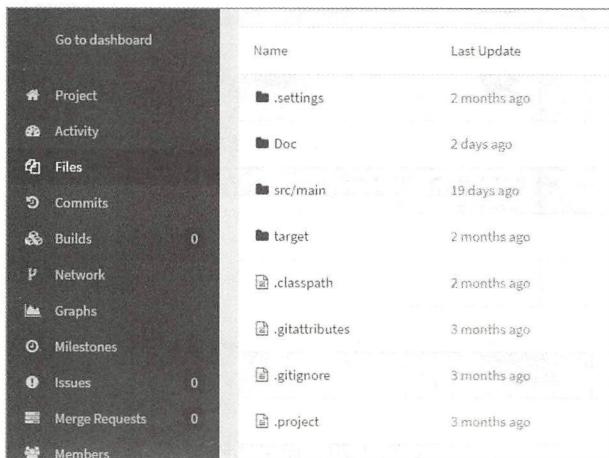


图 4-22 查看提交成功后的代码



4.4 JDeploy 平台

JDeploy 平台是采用 Java 和 Shell 实现的基于 Linux 系统的自动化、可视化的项目部署平台。传统的部署模式是从 SVN 或 Git 服务器检出代码，在 IDE 中编译打包，把编译完成的包上传到应用服务器或者 Web 服务器中，启动应用服务器，完成项目的部署。采用 JDeploy 部署只需要创建项目，在创建项目时选择部署的环境，然后通过 JDeploy 提供的界面一键部署系统，完成项目的启动和停止。如果需要查看日志，可以通过 JDeploy 的界面查看服务器的日志，不需要登录到服务器，便于运维工程师查找错误日志。

4.4.1 认识 JDeploy

前面讲解了自动化部署的流程需要一个自动化部署平台的支撑，而 JDeploy 就是一个开源的一键式自动化部署平台，改变了传统的部署方法。与其他开源部署工具相比，JDeploy 具有如下特点。

- 支持从 SVN/Git 仓库中获取源代码，不需要手动获取代码。
- 一键式发布到开发环境、测试环境及生产环境。
- 代码开源通过 Java 和 Shell 实现，可以进行二次开发。
- 简化了手工部署的流程。
- 可以快速查看运行情况和运行日志。

4.4.2 JDeploy 部署配置

下面简单介绍 JDeploy 的配置步骤。

(1) 下载 JDeploy 源代码，其在 github 的下载网址为 <https://github.com/wucaoj/JDeploy>。

(2) 创建 JDeploy 所依赖的数据库。新建数据库 JDeploy 并导入刚才下载完成的 sql.sql 文件。

(3) 修改 Config 配置文件，包括设置数据库连接信息、Shell 脚本位置、Tomcat 位置和项目部署位置。

配置文件 Config.properties 位于 src/main/resources/目录下。



```
# 数据库配置
jdbc.url=jdbc:mysql://localhost/Jdeploy?useUnicode=true&characterEncoding=UTF-8
jdbc.username=root
jdbc.password=123456

# Shell 脚本位置
shell.javadeploy=doc/shell/javadeploy
shell.javawebdeploy=doc/shell/javawebdeploy

# Tomcat 位置
javawebdeploy.tomcatpath=/usr/local/apachet-tomcat-8.8

# 项目部署位置
javadeploy.basepath=/root/Desktop/jdeploy/java
javawebdeploy.basepath=/root/Desktop/jdeploy/javaweb

# 功能模块
modules=java,javaweb
```

(4) 设置登录用户名和密码。用户名和密码使用 SpringSecurity 管理，需要修改配置文件 SpringSecurity.xml，修改 user 节点中属性 name 和 password 的值。

```
<authentication-provider>
  <user-service>
    <user name="admin"
      password="1234"
      authorities="ROLE_USER" />
  </user-service>
</authentication-provider>
```

(5) 执行 Maven 命令“mvn install”进行打包操作，把打包完成的 War 文件放入 Tomcat 中，启动 Tomcat 完成 JDeploy 部署。

4.4.3 一键部署项目

JDeploy 平台已经部署完成，登录后可以进行项目的发布，包括代码的获取、编译、打包和部署，下面介绍 JDeploy 平台的操作。

(1) 在浏览器的地址栏中输入“http://127.0.0.1:8080/JDeploy”，打开 JDeploy 平台登录界面，如图 4-23 所示。



图 4-23 JD Deploy 登录界面

(2) 输入用户名和密码，进入部署平台主页面，如图 4-24 所示。在此页面中可以部署纯 Java 项目和 Java 的 Web 项目。



图 4-24 JD Deploy 部署平台主页面

(3) 在主页面中单击“创建”按钮后，显示新建项目页面，如图 4-25 所示。输入项目名称、SVN 或者 Git 地址，选择要发布的环境，如 FAT、UAT 或者 PRD，完成后单击“提交”按钮。

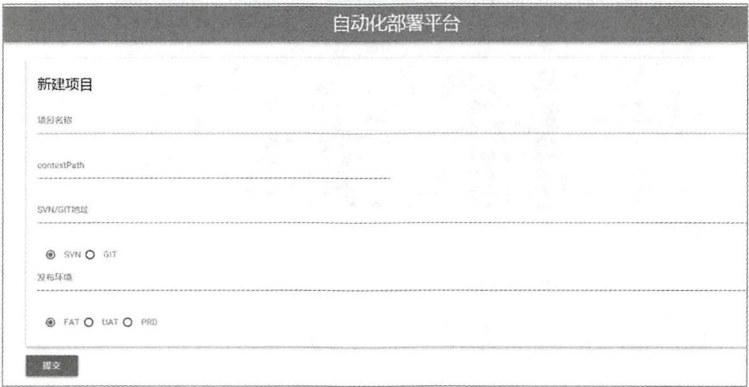


图 4-25 新建项目页面

(4) 返回主页面，选择刚创建的项目，单击“详情”按钮可以查看项目部署的环境和日志，并且可以对项目进行部署、重启和停止操作，如图 4-26 所示。



图 4-26 查看项目运行状态

(5) 单击“查看日志”，弹出日志显示窗口，如图 4-27 所示，通过该窗口可以看到服务器的日志信息，方便运维工程师查找问题。

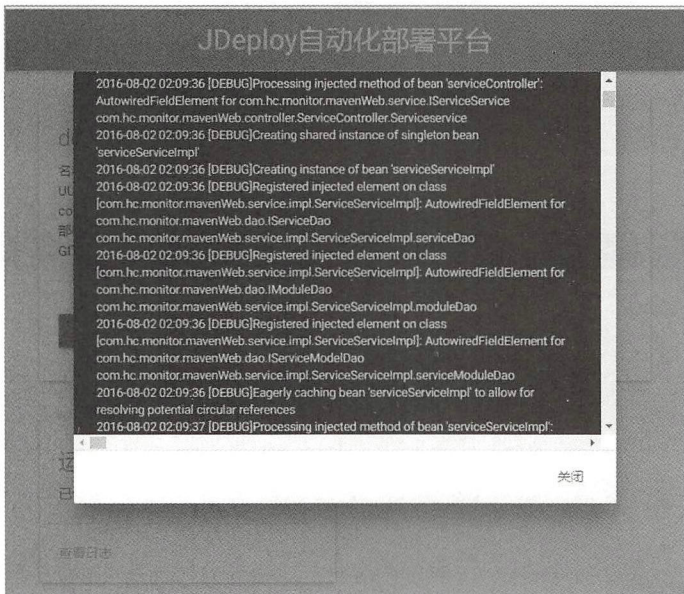


图 4-27 查看项目运行日志

4.5 要点回顾

本章讲解了手工部署的不足和自动化部署的优点，并介绍了自动化部署的一些方法。通过持续集成完成代码的获取、编译、发布和测试相关任务，帮助用户高效地开发软件产品。同时，借助版本控制软件 Git 对代码的版本进行控制和管理，及时了解团队中其他成员的进度，比较不同版本之间的细微差别；记录每个文件更改的细节和历史记录，利于成果的复用；资源共享，避免以往复制文件造成的版本混乱；协同工作，极大提高团队的工作效率。

通过 JDeploy 部署平台实现 Java 项目自动化发布，包含纯 Java 项目和 Java Web 项目。JDeploy 同时也是一款开源的自动化部署平台，可以根据具体业务需要，在 JDeploy 平台基础上进行二次开发，设计出一套符合用户需求的自动化部署平台。

由于每个系统的配置信息不相同，所以自动化部署还需要一个配置中心。配置中心的主要任务是完成项目的配置，使配置系统化，将部署系统和配置中心相结合完成项目的部署。

第 5 章

软件测试

从狭义上讲，软件测试用于确认软件的质量，一方面是确认软件做了所期望的事情，另一方面是确认软件以正确的方式来做这个事情。

从广义上讲，软件测试不仅是在测试产品本身，而且还测试软件开发生命周期的过程。如果一个软件产品开发完成之后发现了很多问题，则说明此软件开发过程很可能是有缺陷的。因此，软件测试是完善和提升软件开发过程的质量的关键。

本章内容主要包括以下几点：

- 功能测试的基本流程；
- 需求原型是什么；
- 根据 UI 设计测试用例；
- 提交缺陷报告；
- 软件测试策略；
- 测试经验分享总结。

5.1 软件测试

软件测试是一个新兴的行业，必须要了解它的现况、发展、未来趋势和各种理论，只有这样才能使之更好地发展，逐步走向成熟。

5.1.1 软件测试发展史

软件测试是伴随着软件的产生而产生的。早期的软件开发，其规模都很小，复杂程度低；软件开发的过程混乱无序，相当随意；测试的含义比较狭窄，开发人员将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分工作；对测试的投入极少，测试介入也晚，常常是等到形成代码、产品已经基本完成时才进行测试。

直到 1957 年，软件测试才开始与调试区别开来，作为一种发现件缺陷的活动。由于一直存在着“为了让大家看到软件在工作，就得将测试工作往后推一点”的思想，潜意识里对测试的目的就理解为“使自己确信产品能工作”。测试活动始终晚于开发活动，测试通常作为软件生命周期中最后一项活动而进行。当时也缺乏有效的测试方法，主要依靠“错误推测”，就是随意操作来寻找软件中的缺陷。因此，大量软件交付后仍存在很多问题，软件产品的质量无法保证。

到了 20 世纪 70 年代，这个阶段开发的软件仍然不复杂，但人们已开始思考软件开发流程的问题，尽管对“软件测试”的真正含义还缺乏共识，但这个词条已经频繁出现。一些软件测试的探索者们建议在软件生命周期的开始阶段就根据需求制定测试计划，这时也涌现出一批软件测试的宗师。每个行业都有杰出的精英成为开天辟地的先驱者，由这些精英开创这个行业，定义一些基础理论。

Bill Hetzel 博士就是软件测试行业的先驱者。1972 年，Bill Hetzel 博士（代表作《软件测试完全指南》）在美国的北卡罗来纳大学组织了历史上第一次正式的关于软件测试的会议。

1973 年，Bill Hetzel 博士给软件测试赋予一个这样的定义：“建立一种信心，认为程序能够按预期的设想运行”。在 1983 年他又将定义修订为：“评价一个程序和系统的特性或能力，并确定它是否达到预期的结果。软件测试就是以此为目的的验证过程。”在他的定义中，“设想”和“预期的结果”其实就是现在所说的用户需求或功能设计，他还把软件的质量定义为“符合要求”。

Bill Hetzel 博士思想的核心是：测试方法是试图验证软件是“工作的”。所谓“工作的”就是指软件的功能是按照预先的设计执行的，以正向思维，针对软件系统的所有功能点，逐个验证其正确性。

软件测试行业把这种方法看做是软件测试的第一类方法（软件测试是功能验证的过程）。

尽管 Bill Hetzel 博士是软件测试的先驱者，但这个方法还是受到很多业界人士的质疑

和挑战。代表人物是 Glenford J. Myers（代表作《The Art of Software Testing》）。Glenford J. Myers 认为测试不应该着眼于验证软件是可以工作的，相反，应该首先认定软件是有错误的，然后用逆向思维去发现尽可能多的错误。他还从人的心理学的角度论证，如果将“验证软件是可以工作的”作为测试的目标，则非常不利于测试人员发现软件的错误。于是，Glenford J. Myers 于 1979 年提出了他对软件测试的定义：“测试是为发现错误而执行的一个程序或者系统的过程”。Glenford J. Myer 认为，一个成功的测试必须是发现 Bug 的测试，不然就没有价值，还给出了与测试相关的三个重要观点。

（1）测试是为了证明程序有错，而不是证明程序无错误。

简单理解：测试是程序的执行过程，目的在于发现错误；不能证明程序的正确性。

（2）一个好的测试用例在于它能发现至今未发现的错误。

（3）一个成功的测试是发现了至今未发现的错误的测试。

简单理解：测试无法证明错误不存在，只能证明软件错误已出现。

这就是软件测试的第二类方法（测试是为了验证软件是有错误的）。简单地说就是验证软件是“不工作的”，或者说功能是无法实现的。这就如同一个病人，到医院做一项检查，结果各项指标都正常，那说明该项检查对于诊断该病人的病情是没有价值的、是失败的。Glenford J. Myers 提出的“测试的目的是证明软件是有错误的”这个概念，推翻了过去“为证明软件正确而进行测试”的错误认识，为软件测试的发展指出了方向，软件测试的理论、方法在之后得到了长足的发展。第二类软件测试方法在业界也很流行，受到很多学术界专家的支持。

然而，“测试的目的是证明软件是有错误的”这样的定义也有一定的片面性，带来一定的后果。若测试人员以发现缺陷为唯一目标，而很少去关注系统对需求的实现，那么测试活动往往会存在一定的随意性。例如，测试提出“界面不美观，按钮排版不统一，输入框无法输入 100 个汉字”，这样的 Bug 其实是没有技术含量的。如果开发人员与测试人员每天因为这样的问题吵来吵去，那么直接的后果就是测试周期延长，测试成本增加，项目交付延期。间接的后果更可怕，开发人员认为测试是没有意义的，因为测试人员发现的问题根本就不是需求里面提到的，用户难以重现，提出的 Bug 都是无足轻重的，那么研发团队以后的核心力和竞争力会大幅度下降。

如果接受了这样的定义，以 Bug 数量作为考核测试人员绩效的重要指标也不太科学。在 2008 年之前有相当多的企业就是这样考核测试人员的，指标就是 Bug 数量，而不关注



软件本身的质量。在当时活跃的一些软件测试论坛中，就有测试人员反映，很多企业测试人员的奖金和 Bug 数量是挂钩的。

提示：为什么是 2008 年？因为这个特殊的年份正是中国软件企业通过 CMMI 认证数量最多的一年。认证机构指出 Bug 数量绝不能作为考核测试人员的标准。

总体来讲，第一类测试可以简单抽象地描述为这样的过程：在设定的环境下运行软件的功能，将测试结果与用户需求相比较，如果相符则测试通过，如果不相符则视为 Bug。这个过程的终极目标是将软件的所有功能在所有设定的环境中全部运行并通过。在软件测试行业中一般把第一类方法奉为主流和行业标准。第一类测试方法以需求为本，因此有利于界定测试工作的范畴，更便于部署测试的侧重点，加强针对性。这一点对于大型软件的测试，尤其是在有限的时间和人力资源的情况下显得尤为重要。

第二类测试方法与需求和设计没有必然的关联，更强调测试人员发挥主观能动性，用逆向思维方式，不断思考开发人员理解的误区、不良的习惯、程序代码的边界、无效数据的输入及系统的各种弱点，试图破坏系统、摧毁系统，目标就是发现系统中各种各样的问题。这种方法往往能够发现系统中存在的更多缺陷。

到了 20 世纪 80 年代初期，软件行业进入了大发展时期，软件趋向大型化、高复杂度，软件的质量越来越重要。这个时候，一些软件测试的基础理论和实用技术开始形成，并且人们开始为软件开发设计了各种流程和管理方法，软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计及结构化测试为特征。人们还将“质量”的概念融入其中，软件测试的定义发生了改变，测试不单纯是一个发现错误的过程，而且是软件质量保证（SQA）的主要组成部分，Bill Hetzel 在《软件测试完全指南》一书中指出：“测试是以评价一个程序或者系统属性为目标的任何一种活动。测试是对软件质量的度量。”这个定义至今仍被引用。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题，软件测试行业开始有了自己的行业标准（IEEE/ANSI）。

1983 年，IEEE 提出的软件工程术语中给软件测试的定义是：“使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别”。这个定义明确指出软件测试的目的是为了检验软件系统是否满足需求，它再也不是一个一次性的或者只是开发后期的活动，而是与整个开发流程融合成一体。



软件测试的工作范围从最初的发现 Bug 到控制软件的质量风险。因为测试是没有办法提高软件质量的，只能反应软件的质量，所以软件的质量更多的需要靠前期的设计和后期的运维。测试发现的 Bug 越多，说明软件质量越差，项目按期交付的风险就越高。至此，软件测试已发展成为一个行业，需要运用专门的方法和手段由专业人员来承担。

第一款应用于 PC（在 MS-DOS 上）的商业测试工具 Autotester 由 AutoTester 公司发布，该公司总部设在得克萨斯州的达拉斯。

1986 年，Paul E.Rook 发表于《IEEE 软件工程学报》的文章《控制软件项目》中，介绍了 V 模型。当时 Paul 在伦敦的 GEC 软件有限公司工作。V 模型表明了在一开发生命周期中各个阶段和相关测试阶段的关系。

1986 年，David Gelperin 和 William Hetzel 一起创立了软件质量工程（SQE）。

1988 年，软件工程研究所（SEI）的软件过程项目的创始人 Watts Humphrey 在 IEEE 论文《描述软件过程》中提出了软件能力成熟度模型。

1989 年，Amnon Landan 和 Arye Finegold 在加利福尼亚州创立了 Mercury Interactive 公司，该公司发布了很多测试自动化产品，例如 LoadRunner。

1995 年，Mercury 公司发布操作记录和回放工具 WinRunner。

1996 年，Kent Beck、Ron Jeffries 和 Howard G. Cunningham 开始第一个极限编程项目，测试行业向单元测试前进一大步。

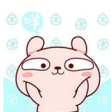
1996 年，测试成熟度模型（TMM）在伊利诺伊理工学院被开发出来，这标志着软件测试走向成熟。

2002 年，国际软件测试认证委员会在爱丁堡成立。

2005 年，各个大公司开始推行 CMMI，最重要的原因是美国和日本外部市场比较大，国外企业相信证书，所以 Infosys 公司、摩托罗拉公司和东软集团很早就通过了 CMM 认证，到 2005 年其他公司都趋之若鹜。

5.1.2 软件测试的未来

软件测试的对象是软件开发过程中所产生的项目立项报告、需求规格说明书、概要设计说明书、详细设计规格说明书及开发的代码。预测人员检查软件或者平台是否满足需求是测试的目标。



未来的趋势对于产品质量来说,预防问题比发现问题本身更重要。把测试工作融入到开发过程中,能减少开发人员的出错机会,还可以极大地减少测试人员上报无效 Bug 的数量。测试人员的主要工作是提供测试的工具和框架。

提示:笔者非常崇拜托尼·霍尔,因为这位前辈在1980年就已经获得美国计算机学会(ACM)设立的计算机界最高奖——图灵奖。他曾经讲过,软件测试的真正价值并不体现在代码中找出多少缺陷,而是发现设计和编程人员解决问题方法上的局限、思路中的狭隘和技能方面的不足。而现在 Google 和 Facebook 都是这么实施的。

5.1.3 测试部门组织架构

软件测试部门是互联网公司的重要职能部门,根据测试类型的不同可以划分为几个不同的小组,如图 5-1 所示是一个标准的测试部门组织架构。

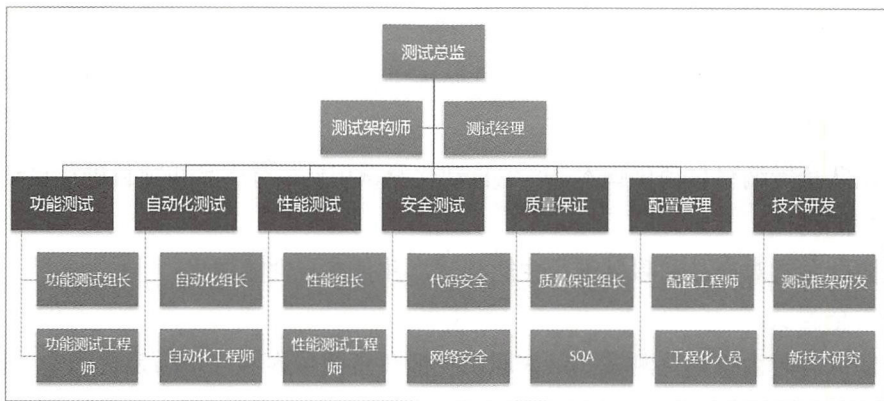


图 5-1 测试部门组织架构

(1) 测试总监。

测试总监根据公司产品的发展方向,及时追踪、收集软件测试新技术、新动态的资料,通过技术培训、交流等方式建立测试团队,提高测试团队的技术和业务能力;优化和完善测试团队,建立符合部门需求的内部管理体制,并进行跨部门协调等工作。

简而言之,测试总监就是负责定义规范、管理团队、向高层汇报。

(2) 测试经理。

测试经理主导软件产品的测试需求分析,负责测试计划和测试方案的制定;预先评估项目的风险并能提出有效规避方案;评审架构和产品设计,给出反馈意见。简而言之,测



试经理就是负责各个待测试项目的总体管控，分别给出指导意见。

（3）测试架构师。

测试架构师对不同产品的测试质量和自动化效率负责，要理解产品目标，提炼测试目标和风险点，提出解决方案、风险预警和预防措施，推动测试、开发、产品团队共同改进；设计并实现相应的工具、平台和解决方案，不断提升产品质量和研发效率。简而言之，测试架构师就是负责提高测试部门的测试水平，尤其在自动化和性能测试方面。

提示：测试架构师的职位是 1999 年在微软公司首先设立的，是专门为那些对产品有影响力的特殊测试贡献者而设立的。测试架构师是一种角色而不是一个职位。尽管一个高级测试工程师可能会晋升为测试架构师，但并不是所有的人都会成为测试架构师。微软公司的测试架构师不仅能够有效地影响测试领域，还能够在开发和项目管理方面发挥影响力。测试架构师必须能驾驭产品的质量，提供指导、反馈和建议，以提高整个工程部门的质量规范。截至 2008 年，微软公司在全球的 9000 多名测试工程师中只有 40 多位测试架构师。

（4）功能测试。

这个团队人数最多，因为公司的项目都必须进行功能测试，分为不同的生产线和产品。

（5）自动化测试。

这个团队负责编写自动化脚本，产生自动化测试用例，对产品和项目进行自动化测试。

（6）性能测试。

这个团队负责对产品和项目进行性能测试，并且参与性能调优。

（7）安全测试。

这个团队有时由自动化测试团队兼职，一般负责两项内容，一项是代码安全检查，另一项是安全漏洞。

（8）质量保证。

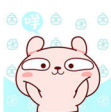
这个团队负责项目流程的制定、管理和监控。关注点是流程是否符合规范，文档是否完备，上下游是否承接顺畅。

（9）配置管理。

这个团队负责定义基线，完成各个版本的备份，在上线日发布最终版本到生产环境。

（10）技术研发。

这个团队负责研究最新的技术或者框架，将这些前沿的信息或者技术应用于实际项目，找出最优方案，最终把科技转化为生产力。



5.1.4 软件测试的基本类型

根据测试类型的不同，软件测试可以分为功能测试、自动化测试、界面测试、性能测试、安全测试、兼容性测试、安装测试、部署测试和可用性测试等，大约有 40 多种类型。

根据测试阶段的不同，软件测试可以分为单元测试、集成测试、系统测试和验收测试。

单元测试是对最小的可测试软件元素（单元）实施的测试。它所测试的内容包括单元的内部结构（如逻辑和数据流）及单元的功能。使用白盒测试方法，验证标准的输入数据和输出结果。

集成测试将所有功能模块按照设计要求组装成系统，进行全部流程的功能验证。

系统测试主要测试系统是否符合《软件需求规格说明书》，进行功能、性能、安全、兼容性等类型的测试。

验收测试主要由项目经理主导，用户根据《用户需求规格说明书》进行验收。

还有一些 UI 测试、耦合测试、模块测试、稳定性测试等都是由不同的公司、不同的测试人员提出的，具有很大的不确定性，没有参考意义。因为这些测试类型可以包含到几个大的测试类型里面，还有一些是标新立异提出的，例如跳转测试，根本没有必要了解。

因此，一个项目关键的测试类型无非就是功能测试、性能测试、自动化测试和安全测试。其中的重中之重就是功能测试，因为软件或者平台最起码的功能一定要实现。性能测试在互联网公司同样是必测类型，因为现在的互联网应用都像秒杀，大量用户要在 1 分钟内登录，所以性能测试与功能测试一样重要。

至于自动化测试，它的根本目的不是为了发现 Bug，而是用于回归测试。例如，一个页面需要用户填写大约 50 项信息，手工录入会比较费时，但是如果编写自动化脚本就会在几分钟内完成。另一个例子是 A 功能上线后，长时间没有更改，现在要上线 B 功能和 C 功能，那么本次上线节点 A 功能进行自动化验证，B 功能与 C 功能进行人工验证，就可以节省人力成本与时间。

最后还要明确一点，就是“验证”和“确认”这两个概念的区别，相当多的测试人员还是会混淆。

- 验证：从开发方的角度，检查最终的产品是否符合需求说明。
 - 在软件生命周期的各个阶段，对阶段产出的成果进行检查或评价以确保满足各自的需求。
 - “验证”强调对需求的满足。



- 确认：从用户的角度，检查最终的产品是否能满足用户的需求。
 - 以用户使用的方式对软件产品或系统进行检查或评价以确定满足用户的需求。
 - “确认”强调客户参与，更直接提升客户满意度。

5.2 测试模型

软件测试就是在软件投入运行前，对项目立项报告、软件需求分析、设计规格说明和编码进行最终审查，是软件或者产品研发过程中的关键步骤。不同的项目类型有不同的测试模型，尤其是在传统的软件公司和互联网企业。本节主要介绍瀑布模型和敏捷模型。

5.2.1 瀑布模型

瀑布模型严格地把软件项目分成各个阶段：需求定义、概要设计、详细设计、编码、单元测试、集成测试和系统测试等。该模型使用里程碑的方式，严格定义了各个开发阶段的输入和输出。瀑布模型更像是生产流水线，如果上一个阶段的输出达不到要求，那么下一个阶段的工作就不能展开。在瀑布开发模型中，开发人员与测试人员工作相互独立，测试人员只有在开发人员交付具有可测性的版本后才会启动测试工作，整个项目的周期很长。

瀑布模型与V字模型的思想非常相近。宽泛地讲，V字模型是瀑布模型的另一种表现形式，重点反映了测试活动与分析和设计的关系，如图5-2所示。

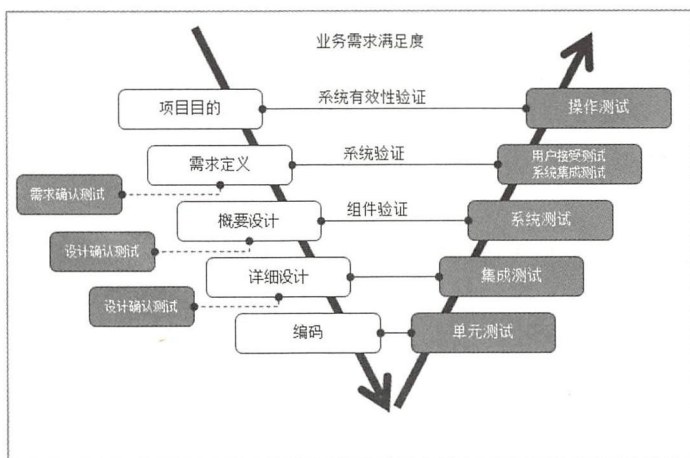


图 5-2 V 字模型



5.2.2 敏捷模型

在 2010 年之后，Google、Facebook 等大量的互联网公司飞速发展，在短短几年间成为行业的翘楚，它们所推崇的敏捷开发模式得到 IT 行业的认可，逐渐被国内的一些互联网公司采用并推广。

敏捷开发模式采用敏捷测试的方法，如图 5-3 所示为敏捷测试流程图。

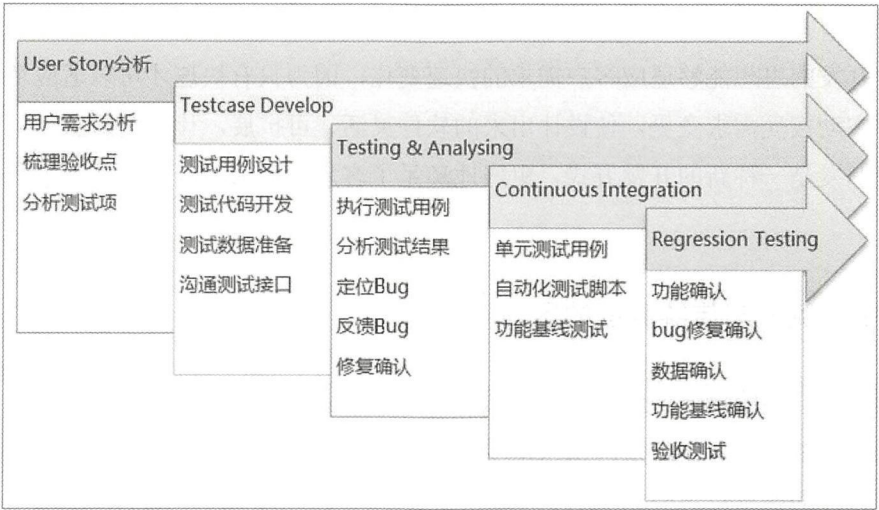


图 5-3 敏捷测试流程图

- User Story 分析：敏捷测试是不断地确认客户的需求，因此对用户需求的分析、理解需要一直持续下去，发现有偏差及时纠正，及时设置合理的验收点和测试项。
- Testcase Develop：设计测试用例，完成测试代码的开发、测试数据的准备，并及时与开发人员沟通软件接口，确保测试代码能够成功驱动业务代码。
- Testing & Analysing：执行测试，统计测试覆盖率，分析测试结果，若发现 Bug 及时沟通，并协助定位 Bug。
- Continuous Integration：将测试代码进行集成，以保证当前功能若被后续集成代码污染能够及时报警，不断地完善软件产品的功能基线。
- Regression Testing：在完成全部 User Story 分析后，对所有代码进行完整的回归测试，对所有 Bug 修复情况进行确认。



质量全面管控：从项目管理到容灾测试

Sprint 是短距离赛跑的意思，敏捷开发以冲刺（Sprint）的形式在最短的时间里完成一次计划的迭代，包括计划、实施、测试和评审。在一个 Sprint 中，测试人员的工作内容主要分为五个部分：User Story 分析、Testcase Develop、Testing & Analysing、Continuous Integration 和 Regression Testing。这五个部分的工作均要持续到 Sprint 结束，只是启动时间有早有晚。

提示：敏捷功能测试 = 新特性的手工测试（针对 User Story 验证） + 原有功能的自动化测试（回归测试）。

敏捷开发的思想能够适应客户需求的快速变化，因为只有快速才可以适应目前社会的快节奏，主动接受需求变更，使设计出来的软件灵活、可扩展，让客户满意，从而获得成功。敏捷开发是一种新的开发方式，更好地满足了客户的需求，应该说是未来的一个发展趋势。

5.2.3 敏捷测试与传统测试的区别

敏捷测试与传统测试的区别并不是敏捷测试测得更快，也不是用的时间更少，更不是将测试的范围缩小或者将质量降低来减少测试任务，而是在计划、阶段划分、文档、记录和沟通等方面的侧重点不同。

（1）传统测试强调测试的计划性，认为没有严谨的测试计划或者不按测试计划执行，测试工作就难以控制和管理。而敏捷测试更强调测试的速度和适应性，侧重测试计划的不断调整以适应需求的变化。

（2）传统测试具有阶段性，从需求评审、设计评审、单元测试、集成测试和系统测试等，到测试计划、测试设计，再到测试执行、测试报告等，在一定的阶段性；而敏捷测试更强调持续测试、持续的质量反馈，模糊了阶段性，而且介入得更早。

（3）传统测试强调发现任何的 Bug 都要记录下来，以便对缺陷的根本原因进行分析，达到预防的目的，并强调缺陷跟踪和处理的流程，区分测试人员和开发人员各自不同的责任；而敏捷测试强调面对面的沟通、协作，强调团队的责任，不太关注对 Bug 的记录与跟踪。

（4）传统测试更关注 Bug，围绕 Bug 开展一系列的活动，如 Bug 的跟踪、度量、分析、报告和质量检查等；而敏捷测试更关注产品本身，关注可以交付的客户价值。在快速交付



的敏捷开发模式下，Bug 修复的成本很低。

(5) 传统测试鼓励自动化测试，但自动化测试的成功与否对测试没有致命的影响；而敏捷测试的基础就是自动化测试，敏捷测试需要有良好的自动化测试手段支撑的快速测试。

(6) 传统测试更强调测试的独立性，将开发人员和测试人员的角色分得比较清楚；而在敏捷测试中，测试人员需要参与全部的开发活动，需要参与整个项目组的所有会议，使之发挥更大的作用。

(7) 测试人员与开发人员的比例问题。在传统的软件公司，以微软公司为例，测试人员与开发人员的比例一般为 1 : 1，而互联网公司如谷歌公司的测试人员与开发人员的比例则为 1 : 10。

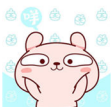
提示：为什么两家公司的差异如此巨大呢？最主要的原因是两家公司对测试人员与开发人员工作范围的定义不同。在微软公司，单元测试由测试人员做，相当于再写一套代码来测试由开发人员写的产品代码，其工作量不比开发人员低（这也是微软公司的测试工程师比研发工程师薪水高的一个原因）。而 Google 公司的单元测试和功能测试一般都是由开发人员自己来完成的，测试人员主要提供自动化测试工具的支持，像性能测试、负载测试和安全性测试等都是由自动化工具来完成的，因此只需要较少的测试人员。所以，经常可以看到 Google 发布一些开源的测试框架，例如 Android 平台自动化测试框架 Espresso。

5.3 系统测试流程

系统测试是整个软件测试生命周期中最重要的步骤，相当多的测试人员或者项目经理都会认为系统测试是在开发编码过程后进行的，这是重大误解。系统测试在项目立项阶段就应该参与。如表 5-1 所示是对整个系统测试过程进行的详细梳理。

表 5-1 系统测试过程

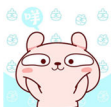
序号	测试执行过程	输出
1	填写《测试需求说明》。 软件测试部门根据所要测试软件产品的情况填写《测试需求说明》，描述计划测试的起止时间、测试所需环境（包括软件、硬件）、测试重点及注意事项、需要进行的测试类型等，以便测试部门安排测试时间与资源分配	《测试需求确认书》



质量全面管控：从项目管理到容灾测试

续表

序号	测试执行过程	输出
2	<p>需求阶段测试。</p> <p>评审需求文档后，填写《需求分析报告》，反馈给需求分析人员和项目经理，可在需求评审会上进行讨论，然后需求分析人员在评审记录表中填写处理结果。测试需求的另一个作用是更好地做计划，测试需求越详细，测试的功能点的数量就越清晰。这部分工作要在前期做好相关准备，如熟悉同类型产品和相关技术资料</p>	<p>《需求评审标准》</p> <p>《需求分析报告》</p> <p>《需求跟踪矩阵》</p>
3	<p>设计阶段测试。</p> <p>在概要设计和详细设计阶段，测试部门会对总体架构、核心数据库设计、用户界面设计等几个方面进行测试。可在概要设计和详细设计的评审会议上提出可行性或者建设性的意见</p>	<p>《设计评审标准》</p> <p>《设计分析报告》</p>
4	<p>测试计划。</p> <p>(1) 项目测试负责人(TL)依据项目的进展情况，参照《测试计划模板》编写《系统测试计划》，主要内容包括测试内容、错误等级描述、测试环境、选用的测试工具、资源分配、测试进度及测试输出等。</p> <p>(2) 系统测试需在《测试需求说明》中描述的系统运行环境下进行，如果条件允许，还应低于和高于系统运行环境的条件进行测试。如果无法搭建适当的测试环境或有特殊要求，需在《系统测试计划》中重点说明。</p> <p>(3) 根据具体项目对进度、成本、质量的不同要求，测试组的介入时机也有所不同，需在项目策划阶段由项目经理与测试负责人共同确定。</p> <p>(4) 对于产品升级项目，应考虑对前一个版本的遗留问题着重进行验证。需将上一个版本的“遗留问题分析表”作为本次测试《升级测试计划》的附件，并在其中记录对遗留问题的测试结果，升级测试主要包括功能测试、性能测试和配置测试（服务器配置和数据库配置）。其中，测试重点是原功能与新功能的耦合部分，这是最容易出现错误的地方</p>	<p>《系统测试计划》</p>
5	<p>测试设计。</p> <p>为确保测试能够在规范、全面的基础上验证系统的准确性和完整性，项目测试负责人要组织人员进行测试设计</p>	<p>《测试用例》</p> <p>《测试用例评审报告》</p>
6	<p>可测性评估。</p> <p>为提高测试效率，测试部门要对软件开发部门提交的测试软件产品预先进行可测性评估，若所提交的软件产品存在以下情况之一，则认为没有可测性，返回开发部门：</p> <p>(1) 所提交的软件产品问题覆盖率非常广，每个功能模块均有不同程度的问题或每千行代码 Bug 数量超过 50 个；</p> <p>(2) 关键模块的重要功能没有实现，严重影响其他接口模块或后续模块的运行；</p> <p>(3) 关键模块重要功能的性能严重超出指标要求</p>	<p>《版本发布说明》</p> <p>《软件配置和系统配置说明》</p> <p>《可测试性标准》</p>





续表

序号	测试执行过程	输出
7	<p>测试执行。</p> <p>在此期间根据测试计划，可以进行功能测试、性能测试、安全测试、兼容性测试、容量测试和容灾测试等，把发现的 Bug 记录在缺陷管理工具中，便于测试人员与开发人员进行沟通</p>	《测试报告》
8	<p>测试通过准则。</p> <p>(1) 《软件需求规格说明书》中的全部功能需求均已实现；</p> <p>(2) 测试用例全部通过；</p> <p>(3) Mantis 中“A”、“B”、“C”三类错误全部关闭；允许存在“D”类错误，遗留的问题少于缺陷总数的 1%；</p> <p>(4) 如有特殊情况而存在遗留问题，不能通过准则(1)、(2)、(3)，则必须由指定仲裁人进行确认后方可释放</p>	《测试通过标准》
9	<p>交叉测试。</p> <p>交叉测试的目的是模拟真实的客户，可以由其他项目组的成员来做，如测试人员、需求人员、设计人员和开发人员。此类测试进行周期不会超过两天</p>	《交叉测试报告》
10	<p>验收测试。</p> <p>需求人员、客户和测试人员三方进行测试，项目经理进行监控。</p> <p>注意：验收测试一定不要用测试人员使用的环境，要重新搭建测试环境，重新部署产品，这样就可以把这次验收测试当成一次预上线，不单测试功能，还测试了环境、数据和网络等。</p> <p>验收测试的测试用例由需求人员编写，《验收测试报告》由测试部门提供模板，由客户填写</p>	《验收测试报告》
11	<p>最终版本的确认与控制。</p> <p>测试部门对最终的版本要进行监控操作，以保证所提交的版本准确无误（由软件配置管理人员进行最终版本的确认）</p>	发布的最终版本
12	<p>跟踪监控。</p> <p>系统在正式发布后，测试人员会对基本功能和性能进行跟踪监测，标准是在满足 72 小时的条件下，在外部硬件设备和网络稳定的情况下，系统正常运行。</p> <p>要点：针对重要模块进行功能和性能跟踪（需要系统人员配合，提供服务器相关性能指标）</p>	《跟踪日志》
13	<p>测试总结。</p> <p>测试负责人要组织测试组成员对测试工作进行总结，并在广泛听取测试人员意见的基础上，参照《测试总结报告》编制系统测试的工作总结，列出在测试中发现的问题，分析测试的重点内容，总结经验教训</p>	《测试总结报告》



质量全面管控：从项目管理到容灾测试

系统测试阶段的关注点如下。

1. 测试环境

项目组根据系统的运行条件准备测试环境，测试负责人对测试环境要确认以下几点。

- 确认计算机硬件、网络、软件支持环境已满足所测试软件的要求，并确认这些环境运行正常；
- 消除病毒干扰，使用杀毒软件对测试环境进行病毒检测和杀毒处理；
- 测试人员应对测试执行过程中的基线版本加以备份；
- 严格禁止在测试服务器上直接修改代码，避免造成版本混乱。

2. 测试用例库

- 使用 Testlink 作为测试用例的管理工具，统计测试用例覆盖率和测试用例执行率的情况；
- Testlink 可以集成 Mantis，可以统计测试用例关联的 Bug 数；
- 测试用例的所有状态会在 Testlink 的统计图表中显示。

3. 测试工具的选择

测试工具只是软件测试的一种方法，它是提升软件测试质量和水平的有效手段，根据具体项目的实际情况可使用不同的工具。

推荐工具：自动化测试工具 Selenium，性能测试工具 LoadRunner 或者 JMeter。

4. 问题提交

测试人员要认真记录测试执行过程中发现的问题，并将问题描述提交到缺陷管理系统（Mantis），测试执行过程中要遵循以下原则：

- 对重大的错误要保留测试现场，请相关开发人员前来查看，尽快解决问题；
- 对错误现象的描述要准确和详尽，必要时要截图与录屏，尽量使开发人员根据测试人员描述的操作步骤重现该问题；
- 测试负责人登录 Mantis 审核测试人员提交的 Bug，并对 Bug 进行修改、删除、保留或重新打开的处理，在必要时通知开发负责人更新 Mantis 中的 Bug 状态。



5. 测试周会

项目经理主持项目周会，测试负责人提交测试分析报告，包括重要缺陷、建议和需要协调的问题。

周会的输出文件是《测试分析报告》。

6. Bug 分类标准

测试中发现的问题按其软件系统影响的严重程度分为以下四类。

● A：严重影响系统运行

关键功能没有实现，影响其他系统的工作，或性能严重超出指标要求，系统反应速度极低，或极限测试时出现系统内存暴涨、CPU 占用率陡增，甚至系统死机，或并发测试时并发量严重低于系统要求。

● B：影响系统运行

具体功能没有实现，影响后续系统的工作，性能测试达不到需求规格说明书中的性能要求。

● C：不影响系统运行，但必须修改（多指可用性方面）

不影响系统运行，但不符合客户术语和公司编码规范，属于必须修改的问题或可用性方面的问题。

● D：所提建议

在需求分析、设计文档中都没有说明，但在实际操作过程中能提高系统的高效性、强壮性和可用性等方面的合理化建议。

根据不同项目的特点，在系统测试计划中对 Bug 分类标准要有详尽的描述。

5.4 根据需求原型设计测试用例

需求原型是需求文档的最简洁说明，客户的要求可以通过需求原型展示，测试人员通过需求原型编写测试用例。需求原型能降低需求理解难度，更有助于理解客户的需求。

5.4.1 需求原型规范样式

在《软件需求规格说明书》评审完成之后，需求人员会根据说明书设计需求原型。例



质量全面管控：从项目管理到容灾测试

如，一个 ABC 超市订货单需求原型包括用户界面原型（如图 5-4 所示）、用户界面原型相关需求说明（如图 5-5 所示）和用户界面原型相关数据库设计说明（如图 5-6 所示）。测试人员根据需求原型设计测试用例。

订货单转送货单

订货单状态：未收货

订货单编号：2016010415424345

送货单编号：3923576023562856

部门：10 饮料

厂商：000001 上海第一有限公司

订单到达平台时间：2016.01.04 15:42

应到货日期：2016.01.04

订货到取消日期：2004.09.29

订货单差异回复截止时间：2016.01.04 18:00

订货单未税总金额：551.2923

门店：9999 上海陆家嘴店

下单日期：2016.01.03

送货地址：上海市陆家嘴路1号

订单备注：自动订货生产送货单

订单差异回复已由门店商品循环部人员打印，不可进行回复！

订货单详细单据如下：门店人员需要收货7箱，应付金额645.0093元

序号	条码	商品编号	子码	商品名称	型号	规格	包装	箱数	单位	零数	数量	单位	赠品	税率	未税单价	含税单价	未税金额	含税金额	备注	厂商商品编号
1	8920845600943	701097	001	蓝带蓝鸟听啤		350ml	24	1	箱	0	24	罐	0	17%	2.3604	2.7600	56.4123	65.9997		
2	7931743680250	401192	001	百威啤酒听啤		500ml	24	1	箱	0	24	罐	0	17%	2.9915	3.5000	71.8045	84.0060		
3	5900668645194	800162	001	雪花清爽啤酒		1.45l	24	2	箱	0	12	瓶	0	17%	11.5385	13.5000	138.4600	161.9982		
4	2901352845397	201046	001	青岛冰爽啤酒		500ml	12	3	箱	0	18	瓶	0	17%	15.8120	18.5000	284.6221	333.0054		
共4种商品								总计	7	0	78		0				551.2989	645.0093		

请于送货前，确认订单之商品单价及税率与实际送货内容相符；于商品验收后之差异，不予以修改

1. 不同订货单上的商品不要写在同一张送货单上

2. 请务必于到货日期前送货

3. 赠品请标明数量及品种

图 5-4 用户界面原型

common rule:
1. 表单右上方要加上“打印日期: yyyy.MM.dd”, “打印时间: hh:mm:ss”, “页次: 第x页/共Y页”, “打印人: j.jm”
2. 报表字段排列格式follow加上原则: 文字居左对齐, 数字居右对齐
3. 对于报表和表单中的金额和单价, 若是数据来自于收货报告主档, 则显示页面上保留小数点后2位, 否则保留小数点后4位, 对于数据库记录的8位小数的单价, 取四舍五入保留4位小数
4. 数量相关字段显示原则: 小数点后不补零, 例如: 12.000显示为12, 12.003显示为12.003

页面说明:
1. 对于涉及到单个商品的信息, 一定要对“商品唯一号”进行比较, 确定“商品唯一号”也匹配
2. 该页面为html格式
3. 若是user没有订单差异回复权限, 则点“订发货单差异回复”按钮提示弹出页面提示“您没有操作订发货单差异回复的权限”
4. “订发货单差异回复”: 自资料上平台开始算, 8个工作日内若是不回复, 则不可回复, 点“订发货单差异回复”按钮提示弹出页面提示“请在订发货单到达平台8个工作日内回复, 现已超时, 不可回复”
所谓8个工作日内概念的解释: 首先要取得订发货通知单主档, 单据进入平台时间 createdate, 系统当前时间 sysdate, 工作时间同以工作日的9:00-18:00开始算, 中间12:00-13:00忽略, 仍然认为是工作时间
所以有三种情况, 举例说明:
if createdate=5.27 5:00, then when sysdate>5.27 18:00, 不可回复
if createdate=5.27 19:00, then when sysdate>5.28 18:00, 不可回复
if createdate=5.27 12:10, then when sysdate>5.28 12:10, 不可回复
5. 订发货单差异回复若是已经被联华打印 (PO_REPLY_MST.PRINT_DATE is not null), 则不可以再被回复 (其实是不能修改回复), 且显示“订发货单差异回复已由门店商品循环部人员打印, 不可进行回复!”; 否则为空白。
6. 若是订单处于已收货, 过期, 取消状态, 则不能做订发货差异回复, 若点“订发货单差异回复”按钮弹出页面提示信息为“订发货单状态为已收货, 不可进行回复”, “订发货单状态为过期, 不可进行回复”, “订发货单状态为取消, 不可进行回复”, 若是其他异常, 则提示“未知原因, 请与客服人员联系”, 请参考页面
7. 差异回复按钮点击后, 若是满足差异回复的条件, 则父页面跳转到“订发货单差异回复”页面, 并且店别, 部门, 订发货单号都已经预输入完毕, 结果页面已经出来
8.
9. 若是user没有送货单新增权限, 则点“订发货单转送货单”按钮提示弹出页面提示“您没有操作订发货单转送货单的权限”
10. 订发货单转送货单按钮失败原因: 已转过送货单, 订发货单状态为已收货, 订发货单状态为过期, 订发货单状态为取消, 以上四种状态的订发货单的话, 不能做订发货单转送货单, 若点“订发货单转送货单”按钮, 则弹出页面提示信息为“已转过送货单”, “订发货单状态为已收货”, “订发货单状态为过期”, “订发货单状态为取消”, 若是其他异常, 则提示“未知原因, 请与客服人员联系”, 请参考页面
11. 由于同一张订单不能重复产生送货单, 故在产生送货单之前要判断“订发货通知单主档. 送货单号”是否为null, 只有为null的情况下才能转送货单。
12. 税率显示为对应的tax_name
13. 若是用户有开启权限, 则当他看到该页面时, 若是该订单对应的POH_print_at为null, 则记录POH_print_at栏位为sysdate
14. 上周总金额两比率概念, 请参考工作快速说明

图 5-5 用户界面原型相关需求说明



订货通知单主表 (POH), 订货通知单明细表 (POD)			
head 栏位名称	栏位描述	数据库栏位、表格	备注
订单状态		订货通知单主表. 订单状态	根据状态编号, 显示对应中文名称
门店	门店编号-门店名称	订货通知单主表. 门店编号	门店名称关联门店基本资料表
部门	订货部门编号-部门名称	订货通知单主表. 部门编号	部门名称关联部门基本资料表
订货单编号		订货通知单主表. 订货通知单号	
厂商	厂商编号-厂商名称	订货通知单主表. 厂商编号	厂商名称关联厂商基本资料
应到货日期	yyyy.MM.dd	订货通知单主表. 应到货日期	
下单日期	yyyy.MM.dd	订货通知单主表. 下单日期	
订货取消日期	yyyy.MM.dd	订货通知单主表. 订货取消日期	
订货单总金额		订货通知单主表. 未税总金额	保留小数点后4位, 不足部分以零替代
送货地址	门店地址		通过门店编号关联门店地址
订单备注		订货通知单主表. 备注	字体内容比其他字体大两号, 红色粗体字
detail 栏位名称	栏位描述	数据库栏位、表格	备注
序号		订货通知单明细表. 录入商品序号	
条码		订货通知单明细表. 商品条码	
商品编号		订货通知单明细表. 商品编号	
子码		订货通知单明细表. 商品子码	
商品名称	商品唯一号-部门编号-商品编号-子码		关联子商品基本资料表
型号		订货通知单明细表. 商品型号	
规格		订货通知单明细表. 商品规格	
包装	一箱有多少个商品	订货通知单明细表. 包装数量	
箱数	以包装单位为基础的商品数量	订货通知单明细表. 商品数量(包装)	
单位		订货通知单明细表. 包装单位	
零数		订货通知单明细表. 零头	
数量		订货通知单明细表. 商品数量(单品)	
单位		订货通知单明细表. 商品最小单位	
赠品		订货通知单明细表. 赠品量	
税率		订货通知单明细表. 税率	以取得税率值关联税率表(TAX, TAXID), 取得TAX_NAME显示
未税单价		订货通知单明细表. 未税单价	保留小数点后4位, 不足部分以零替代
含税单价		订货通知单明细表. 含税单价	保留小数点后4位, 不足部分以零替代
未税金额		订货通知单明细表. 未税金额	保留小数点后4位, 不足部分以零替代
含税金额		订货通知单明细表. 含税金额	保留小数点后4位, 不足部分以零替代
备注		订货通知单明细表. 备注	
厂商商品编号			关联厂商商品编号反馈表, 若是找不到资料, 则显示为空

图 5-6 用户界面原型相关数据库设计说明

5.4.2 设计测试用例

根据以上用户界面原型, 设计如下功能测试用例。

● 表头验证

表头验证主要验证表头的各字段是存在的, 并且没有多余。设计表头验证测试用例时, 需要参照如图 5-4 所示的用户界面原型。

● 栏位验证

设计栏位验证测试用例时, 需要参照如图 5-4 所示的用户界面原型。

栏位由左到右分别为序号、条码、商品编号、子码、商品名称、型号、规格、包装、箱数、单位、零数、数量、单位、赠品、税率、未税单价(显示到小数点后第四位)、含税单价(显示到小数点后第4位)、未税金额(显示到小数点后第四位)、含税金额(显示到小数点后第四位)、备注和厂商商品编号。

另外, 确认在表格的最后一个栏位分别对商品数(共X种商品)、总计(位置在“包装”栏正下方)、箱数、零数、数量、赠品、未税金额(显示到小数点后第四位)和含税金额(显



质量全面管控：从项目管理到容灾测试

示到小数点后第四位) 进行汇总, 确认计算正确无误。

● 数据验证测试

设计数据验证测试用例时, 需要参照如图 5-6 所示的用户界面原型相关数据库设计说明, 通常由数据库管理员提供。

请至数据库查询(订货通知单主表 POH), 比对字段里的数据是否显示正确, 如表 5-2 所示。

表 5-2 订货通知单 POH

订单状态	订货通知单主表 POH.status
门店	订货通知单主表 POH.vendor_store
部门	订货通知单主表 POH.store_code
订货单编号	订货通知单主表 POH.number
厂商	订货通知单主表 POH.vendor
应到货日期	订货通知单主表 POH.rc_date
下单日期	订货通知单主表 POH.no_date
订货取消日期	订货通知单主表 POH.cancel_date
订货单总金额	订货通知单主表 POH.summary
送货地址	订货通知单主表 POH.address
订单备注	订货通知单主表 POH.remark

请至数据库查询(订货通知单明细表 POD), 比对字段里的数据是否显示正确, 如表 5-3 所示。

表 5-3 订货通知单明细表 POD

序号	订货通知单明细表 POD.sub_number
条码	订货通知单明细表 POD.barcode
商品编号	订货通知单明细表 POD.only_code
子码	订货通知单明细表 POD.sub_code
商品名称	订货通知单明细表 POD.sub_name
型号	订货通知单明细表 POD.seq
规格	订货通知单明细表 POD.capacity
包装	订货通知单明细表 POD.pack
箱数	订货通知单明细表 POD.qty
单位	订货通知单明细表 POD.unit
零数	订货通知单明细表 POD.zero



续表

序号	订货通知单明细表 POD.sub_number
数量	订货通知单明细表 POD.pack_size
单位	订货通知单明细表 POD.ordey_unit
赠品	订货通知单明细表 POD.freeqty
税率	订货通知单明细表 POD.vat
未税单价	订货通知单明细表 POD.net_amt
含税单价	订货通知单明细表 POD.amt
未税金额	订货通知单明细表 POD.amt_price
含税金额	订货通知单明细表 POD.net_price
备注	订货通知单明细表 POD.sub_remark
厂商商品编号	订货通知单明细表 POD.vendor_code

5.5 缺陷描述

软件中的缺陷（Defect 或 Bug）是软件开发过程中的“副产品”。通常 Bug 会导致软件产品在某种程度上不能满足用户的使用。一个产品如果有很多严重 Bug，就会使客户失去信心，不愿意使用。在软件开发生命周期的后期，修复检测到的软件错误的成本较高，所以越早发现缺陷，越可以减少软件开发成本。

在提交缺陷时，测试人员描述不清会影响开发人员对此缺陷处理时的效率，后续重现缺陷时也会有困难。测试人员可以截图或录制视频短片说明这个缺陷确实出现过。

5.5.1 缺陷属性

对缺陷属性的描述应包含如表 5-4 所示的内容。

表 5-4 缺陷属性及其描述

缺陷属性	描述
缺陷 ID	唯一的缺陷 ID，可以根据该 ID 追踪缺陷
缺陷标题	描述缺陷的标题
缺陷的详细描述	对缺陷进行详细描述，以及缺陷重现的步骤等。对缺陷描述的详细程度直接影响开发人员对缺陷的修改，描述应该尽可能详细



续表

缺陷属性	描述
缺陷状态	缺陷常见的状态有：“新建”、“待解决”、“已解决”、“已关闭”、“拒绝”、“重新建”和“延缓”。一般来说测试人员识别缺陷，其初始状态是“新建”；项目经理或技术经理分析缺陷，分配给合适的开发人员来解决，状态流转为“待解决”；指定的工程师解决缺陷，将其状态跟踪到“已解决”；测试人员复核该缺陷，如果复核通过，则关闭缺陷，状态是“已关闭”，如果复核不通过，则返回到“待解决”；缺陷被认为是不合理的或无法重现的，不会被修复，直接“拒绝”；后续其他代码修改时引起同样的缺陷问题会“重新建”；缺陷被挪到下一个版本修复会设置成“延缓”
缺陷的严重程度	缺陷的严重程度一般分为“致命”、“严重”、“一般”和“细微”四种
缺陷的紧急程度	缺陷的紧急程度为 1~4，1 是优先级最高的等级，4 是优先级最低的等级。 缺陷的紧急程度与严重程度虽然是不一样的，但两者密切相关，往往越是严重的就越紧急。不过，有时有些严重缺陷在讨论之后可能发现修复成本比较大，就会延缓到下一个版本修复
缺陷所属项目/模块	缺陷所属的项目和模块最好能较精确地定位至功能模块
缺陷提交人	缺陷提交人的名字
缺陷提交时间	缺陷提交的时间
缺陷指定解决人	在缺陷为“新建”状态时空，在缺陷为“待解决”状态下由项目经理或开发负责人指定相关开发人员修改
缺陷指定解决时间	指定的开发人员修改此缺陷的截止时间
缺陷解决人	最终解决缺陷的人员
缺陷处理结果描述	对处理结果的描述，如果对代码或者数据库进行了修改，则要求在此处体现出修改
缺陷处理时间	缺陷处理完成的时间
缺陷复核人	对被处理缺陷进行复核的验证人
缺陷复核结果描述	对复核结果的描述（通过或不通过）
缺陷复核时间	对缺陷复核的时间
测试环境说明	对测试环境的描述
必要的附件	对于某些用文字很难表达清楚的缺陷，使用图片或视频等附件是很有必要的。如能获取到错误日志（Error Log），也可以附加上去，有助于开发人员解决问题，特别是那种比较难重现的缺陷

除上述缺陷属性外，使用不同的统计方式，还可以添加“缺陷引入阶段”、“平台”、“缺陷修正工作量”、“备注”、“关联的测试用例”和“关联的需求”等属性。

5.5.2 缺陷描述示例

不同的公司有不同的 Bug 书写规范，但核心的要求就一点，要简捷、清楚地反映真实问题，最高的标准就是根据测试人员提交的 Bug，一个不熟悉系统的人可以按照描述重现



场景。下面就根据一个真实的 Bug 为例进行说明，如图 5-7 所示是一个正在进行测试中的订单查询界面，其中有 10 个 Bug，这个图要作为附件编写到 Bug 中。

订货款转送货款

订货款状态: 未收货

订货款编号: 2016010415424345

送货单编号: 3923576023862856

门店: 9999 上海陆家嘴店

下单日期: 2016.01.03

送货地址: 上海市陆家嘴路1号

订货款备注: 自动订货款生产送货单

订货款差异回复已由门店商品循环部人员打印, 不可进行回复!

订货款详细单据如下: 门店人员需要收货7箱, 应付金额645.0093元

ABC超市订货款查询

部门: 10 饮料

厂商: 6000001 上海有限公司

应到货日期: 2016.01.04

订货款差异回复截止时间: 2016.01.04 18:00

订货款到达平台时间: 2016.01.04 18:42

订货款取消日期: 2004.09.29

订货款未税总金额: 645.0917

图 5-7 订单查询界面错误

图 5-7 标记出的 Bug 是以图 5-5 所示的用户界面原型为基准的，具体描述如下：

使用用户名 aaa、密码 1234 登录系统，进入订货款查询页面。

(1) 进入“订货款汇总查询”页面；单击任何一笔订货款，弹出“ABC 超市订货款”，名称应该是“ABC 超市订货款查询”，而且字体居中、加粗。

(2) 表头“厂商”的名称显示不完全。

(3) 表头的栏位名称和数值内容的空格数要相等，上下要对齐。

(4) 栏位内容缺少“含税单价”和“含税金额”两个字段。

(5) 栏位中文字没有居中，例如从序号—商品编号。

(6) 栏位中的商品编号的数值没有靠右，所有表格内容应该是文字靠左，数字靠右。

(7) 订货款未税总金额应该是四种商品未税金额汇总的数据值，而不是含税金额汇总。

(8) 条码 590066864519，发现表中的条码列没有内容也没有显示完整，缺少 1 位，应该 13 位。

(9) 数量汇总值计算错误。

(10) 数量、零数、赠品、箱数、包装数值是整数直接显示整数，若是“0”就直接写“0”，若是小数则显示到小数点后四位。

实际结果：

订货款未税总金额是四种商品含税金额汇总的数据值 645.0917。

期望结果：

订货单未税总金额应该是四种商品未税金额汇总的数据值 551.2989。

以上的缺陷可以提交至 Mantis，类似于如图 5-8 所示。

项目	分类	查看权限	报告日期
OA系统	[所有项目] J---OA系统	公开	2017-02-16 14:24
大使赫			
高	严重性	很严重	出现频率
新建	处理状况	未处理	
	操作系统		程序测试版本
0000158: 订货单错误			
使用用户名aaa，密码1234。登录系统，进入订货单查询页面。			
1、进入"订货单汇总查询"页面；单击任何一笔订货单，弹出"ABC超市订货单"，名称应该是"ABC超市订货单查询"，而且字体居中、加粗；			
2、表头"厂商"的名称显示不完全；			
3、表头的栏位名称和数值内容的空格数要相等，上下要对齐；			
4、栏位内容缺少"含税单价"，"含税金额"两个字段；			
5、栏位中文字体没有居中，例如从序号-商品编号；			
6、栏位中的商品编号的数值没有靠右，所有表格内容应该是文字靠左，数字靠右；			
7、订货单未税总金额应该是四种商品未税金额汇总的数据值，而不是含税金额汇总；			
8、条码590066864519，发现表中的条码列没有内容没有显示完整，缺少1位，应该是13位；			
9、数量汇总值计算错误；			
10、数量，零数，赠品，箱数，包装数值是整数直接显示整数，若是0就直接写0，若是小数则显示到小数点后四位。			
实际结果：			
订货单未税总金额是四种商品含税金额汇总的数据值645.0917。			
期望结果：			
订货单未税总金额应该是四种商品未税金额汇总的数据值551.2989。			

图 5-8 Mantis 缺陷实例

提示：Bug 实例中描述的 10 条错误，根据不同的测试规范可以进行拆分。例如，把一些界面错误拆分为一个 Bug，把“含税金额汇总”、“未税金额汇总”这样严重的计算错误拆分为一个 Bug。

5.6 测试的策略

在项目测试的过程中，项目组一般会组织召开周会，时间通常在周一或者周五下午，而笔者认为周一上午是最好的项目例会时间。周会中汇总上一周的测试进度和状态，提出遇到的问题，罗列重要的缺陷，计划本周的测试进度和人员安排。必要时，用图表报告更有影响力和说服力。

如果是敏捷测试，还会有每日例会，一般 15~30 分钟。每日例会上同样需要汇报测试进度和状态、遇到的问题和缺陷，还有今日的计划，但是要更简短。

当测试进度因为各种原因，如测试环境、人员安排、需求变更和上下游系统等造成延期，一定要向测试经理和项目经理告知其中的风险和问题，尽快地解决。

每一轮测试，如冒烟测试、回归测试都需要出一份测试报告。报告中要罗列出当前测试的总测试用例数、成功测试用例数、失败测试用例数、执行测试用例数、未执行测试用例数；还要列出发现的新、旧缺陷数，按模块、优先级和严重性分类；列出当前测试遇到的问题，写明某些模块或子系统未被执行的原因。测试报告的解读需要依赖一个简明、阅读性强的测试模板，让项目组人员可以迅速明白测试的结果，定位测试中存在的问题。

5.7 测试过程的经验总结

(1) 描述 Bug 时一定要清晰明了，要防止成为研发人员的讲解员。描述 Bug 一定要有详细的操作步骤和预期实际结果，一图胜百言，关键步骤、重要错误和主要操作过程一定要截图；复杂的操作过程尽量录屏。可以想象成测试人员在一个跨国公司，国家和时区不同，那么国外的研发人员可以根据测试人员的描述重现 Bug，而无需 E-mail 或者电话沟通。另外，新建 Bug 时，如果有后台或前端的错误日志，也可以添加附件，便于开发人员更快地定位问题所在，减少 Bug 修复的时间。

(2) 若测试用例的覆盖度过低，就会使得很多功能未测试。在设计测试用例的时候，一定要尽量组合出更大的覆盖范围，比如路径覆盖、边界值。测试用例要考虑正常的输入，也要考虑异常的输入。

(3) 要考虑多个模块之间的整合问题，一个模块的数据流会影响另一个模块的功能表现。数据量和逻辑越复杂的模块越容易受到其他模块的影响。所以，在设计测试用例时，要考虑到多模块之间的不同测试组合，特别是有报表功能时。

(4) 冒烟测试是必不可少的，关键功能模块一定要覆盖。在冒烟测试时，有时会遗漏关键功能模块的验证，以至于进入回归测试时才发现有些模块几乎不能使用。在设计测试用例时，要分优先级和重要等级，按照模块选取分级最高的用例，结合起来就可以组成冒烟测试用例集。

(5) 多个系统进行集成测试时，环境、版本、权限和数据要充分定义和协调，避免在后期执行时，发现有些数据因为权限限制获取不到，又需要重新执行上级系统的测试用例，导入数据。这样做也可避免环境版本不统一，造成已修复的缺陷又重现，或者开发人员无法重现测试人员新发现的缺陷。

(6) 回归测试决定了版本迭代的质量。每一个版本的迭代，都需要执行回归测试来确保原有通过的测试用例依旧正常。缺少了回归测试，会将风险带入生产环境，一旦发生问题，责任将是测试人员的。所以，无论项目周期如何紧凑，都不能同意压缩回归测试的时间、跳过回归测试。回归测试集中的测试用例选用，不仅要考虑高级别的测试用例、所有的冒烟测试，还需要考虑低级别的关键功能点。在设计测试用例时，一般需要对功能测试用例和回归测试用例进行分类。所有的回归测试用例不管高中低等级，都需要在回归测试中执行。

(7) 在项目计划阶段，各个部门需要预估时间，若项目经理发现测试的时间比开发的时间还要长，就会质疑。这是因为测试的时间等于第一轮系统测试时间加上回归测试的时间，而回归测试通常是 3 轮以上，包括测试人员提出问题、研发人员修改问题，可能还要更改小的需求点，所以测试的时间会长一些。减少测试的时间绝对不是通过加班或者增加人员来解决的，这是治标不治本的方法，正确的方法是回归测试时基本功能使用自动化脚本测试，复杂功能进行人工测试。

(8) 周期较长的项目引入自动化有助于节约人力成本。对于重复性的冒烟测试和回归测试，尽量选择自动化，更早地提供系统质量评估。功能测试人员可以从这些重复的简单测试中抽身出来，专注于更复杂的或机器无法模拟的操作，可以有更多的时间参与到人员协调、与其他项目组成员的交流中来。

(9) 测试虽然是最后一个阶段，但是任何优秀的产品都不是测试出来的，靠的是设计和运维，应该从源头控制，狠抓需求和设计，提高运维水平。

(10) 建议开发人员做单元测试，因为这个步骤是成本最少、效果最显著的步骤，同时经验资深的测试人员应该积极参与单元测试。

(11) 测试申请的提出，不能随便一个开发人员或者邮件就指派，应该由项目经理提出。

(12) 性能测试的环境一定要与系统测试环境的配置统一，此处的统一不是硬件数量和硬件配置，而是配置文件和配置参数。

(13) 性能指标一定要由项目经理和业务人员提出，绝对不能由测试人员提出。

(14) 设立共享团队，这个团队可以随时补充性能团队、自动化团队和功能团队。

5.8 质量保证

软件测试人员的职责是尽可能早地找出软件缺陷，确保得以修复。而软件质量保证人

员的主要职责是创建或者制定标准和方法，提高并促进软件开发能力和减少软件缺陷。

测试人员的主要工作是测试，软件质量保证人员的主要工作是检查与评审，而测试工作也是软件质量保证人员的检查和评审对象。

提示：软件质量控制（QC）是满足质量需求的技术和活动，检查工作产物是否正确。

软件质量保证（SQA）是管理已定义的标准、规范、规程、方法和过程能够在项目过程中得到有效应用的一套有计划的系统化手段，检查流程是否被正确执行。

5.9 要点回顾

本章讲解了以下几点内容：

- 软件测试的历史、现状和未来的趋势；
- 测试模型；
- 功能测试的基本流程；
- 根据需求原型设计测试用例；
- 缺陷描述；
- 测试经验总结。

第 6 章

安全测试与安全管理

互联网行业日新月异，主流不再是传统的软件，而是越来越多的互联网网站和网络应用，公司都将应用架设在网络上，为客户提供更方便、快捷的服务。这些应用在功能和性能上都随着技术的日益成熟不断完善和提高，然而在重要的安全特性上，却没有得到足够的重视。这样就造成黑客们将注意力从以往对网络服务器的攻击逐步转移到对 Web 应用的攻击上。根据 IDG 的最新调查，信息安全攻击有 85% 都是发生在 Web 应用上而非网络层面上。同时，数据也显示，三分之二的 Web 站点都相当脆弱，易受攻击，95% 的网站都受到过类似 SQL 注入等安全攻击。

本章介绍在 IT 测试过程中的安全问题，带领读者进入安全测试的领域，认识安全测试，发现安全问题，并对这些安全问题进行分析，找出解决方法。本章的要点如下：

- 安全测试的概念；
- 开发安全规范；
- 安全测试工具的使用；
- 信息收集并制定测试策略；
- 信息收集，注入攻击，安全漏洞；
- 熟悉安全运维管理平台 OSSIM。

6.1 安全测试

在软件开发阶段，特别是接近产品开发完成阶段，就开始引入安全测试以检验产品符

合安全需求定义。接下来详细介绍安全测试的定义、流程和目标。

越来越多的互联网应用上线，网络安全知识的普及和频频发生的安全事件使用户对网络安全的要求越来越高，那么这就要求互联网产品不仅在功能和性能上要满足用户的需求，而且必须确保用户的使用安全，尤其很多涉及收付款的网络应用。在这样的背景下，安全测试就显得异常重要。安全测试首先会对被测试系统做系统分析，分析其架构、软件体系及程序部署等，再对被测系统做系统安全分析，然后对系统进行安全建模，明确本系统可能存在的各个潜在威胁，接着剖析系统，确认有哪些攻击界面，根据测试方案进行测试。

6.1.1 安全测试概述

安全测试是在软件产品开发基本完成时，验证产品符合安全需求定义和产品质量标准的过程。

那么，安全测试和可靠性测试一样吗？一般来说，可靠性测试更侧重于测试系统可以长时间正常工作，从专业角度来说，产品的可靠性越高，产品可以无故障工作的时间就越长。而安全事故是危害程度最大的失效事件。软件的可靠性不能完全取代软件的安全性，因为安全性还包括在非正常条件下不发生安全事故的能力。

安全测试是检查系统对非法侵入渗透的防范能力。在安全测试期间，测试人员采用各种办法试图突破系统防线，并对安全系统进行防御。如表 6-1 所示为常见的安全事故及其描述。

表 6-1 安全事故及其描述

事故类型	危害描述
DDoS 攻击	对生产环境进行拒绝服务攻击，造成服务不可用等
ARP 欺骗攻击	对生产环境网络中的核心交换机造成 CPU 负载过高等
账户入侵渗透	客户账户的资金被窃取
论坛被渗透	利用论坛挂载木马控制服务器，进一步渗透生产网络
交易敏感信息被窃取	相关资金等信息以明文方式传输时被外部窃取使用
生产程序信息泄露	生产环境的相关资料信息被外部知悉，从而被渗透利用
程序版本存在安全隐患	导致生产环境的系统被远程控制

从理论上讲，只要有足够的时间和资源，没有无法进入的系统。因此，系统安全设计的准则是使非法侵入的代价超过被保护信息的价值。

6.1.2 安全测试的基本过程

在讲解安全测试流程之前，先介绍整个安全开发生命周期都有哪些阶段，安全测试充当什么角色。如图 6-1 所示是安全开发生命周期，安全角色贯穿整个培训、需求阶段、设计阶段、构建阶段、验证阶段、发布和响应阶段。



图 6-1 安全开发生命周期

其中，各个阶段的职责和角色如下。

- (1) 培训：成立一个安全小组，组织项目成员参加安全技术培训和安全意识培训，并且制定安全规范，让每一个成员都意识到安全的重要性。
- (2) 需求阶段：在需求收集和分析的过程中，评审安全需求并建立质量标准，识别安全风险。
- (3) 设计阶段：采用安全架构来设计系统，分析攻击面，并且实行威胁建模。
- (4) 构建阶段：选取安全开发工具，将安全开发规范落实到位，开发人员使用安全的 API，遵守安全开发规范，尽量减少非安全代码的引入。
- (5) 验证阶段：使用 Checkmarx 等工具进行代码审查，选用渗透工具进行渗透测试，尽量多地发现安全 Bug，降低产品上线的安全事故风险。
- (6) 发布：在产品发布之前，制定应急响应方案和安全运维规范，并对集成环境进行安全检查，确保上线时产品平滑地运行。

(7) 响应: 当有安全事故发生时, 运维人员执行应急响应方案, 快速解决事故问题; 同时, 结合人工巡查和漏洞扫描、安全情报系统, 一旦出现安全问题, 立即警报, 争取时间。

在整个安全生命周期中, 安全测试人员要对早期的安全规范和需求进行评审, 选择工具并制定策略, 在验证阶段执行渗透测试等。安全测试人员还要进行环境搭建和准备, 再进行信息收集、系统渗透测试, 对系统安全漏洞进行分析和加固, 最后编写并提交安全测试报告, 其主要流程如图 6-2 所示。

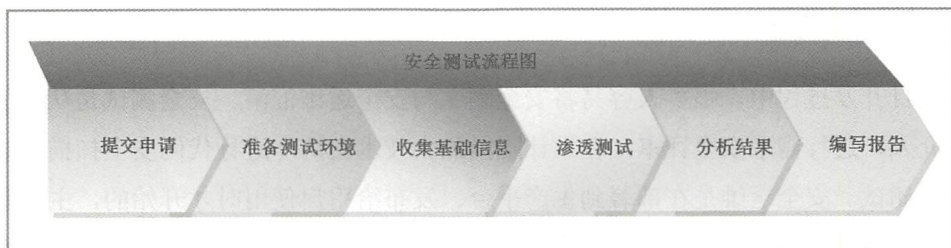


图 6-2 安全测试流程

(1) 提交申请: 成立一个项目组并在内部召开项目启动会议, 以确定此次渗透测试的项目人员和实施此次渗透测试的具体日期及渗透范围。

(2) 准备测试环境: 部署安全测试环境, 包含数据库、应用服务器, 以及路由器、防火墙等网络设施。

(3) 收集基础信息: 针对测试对象进行信息收集活动。信息收集的过程是渗透测试最基础也是最重要的部分, 它为之后的渗透测试工作提供了最基本的信息, 进而缩小了目标范围, 使渗透测试工作简单高效, 减少不必要的麻烦。

(4) 渗透测试: 在已有的信息收集的基础上进行有针对性的测试。此过程是在信息收集的基础上, 进行有先后、有重点的针对性渗透测试。

(5) 分析结果: 在渗透测试结果的基础上进一步分析、挖掘这些漏洞可能导致的其他后果及影响, 检查是否可以利用漏洞, 或是真正地越权取得信息, 抑或是可以进一步利用, 从而取得控制权。

(6) 编写报告: 编写此次渗透测试相关的文档和报告, 项目人员整理渗透测试数据。

安全测试的目标是通过对系统进行精心、全面的脆弱性安全测试, 发现系统未知的安

全隐患并提出相关建议，确保系统的安全性。安全性一般分为应用程序级别的安全性和系统级别的安全性。

- 应用程序级别的安全性：包括对应用数据或业务功能的访问，核实应用程序的操作者只能操作被授权访问的那些功能或数据。
- 系统级别的安全性：包括对操作系统的目录或远程访问，主要核实具备系统和应用程序访问权限的操作者才能访问系统和应用程序。

6.1.3 安全测试与安全运维

在项目开发过程中，很多人容易将安全测试与安全运维混淆。安全测试是从产品立项开始就介入，进行需求安全评审，在设计阶段加大技术支持，进行代码安全扫描、漏洞扫描、渗透测试。安全运维是在部署到生产平台、发布给用户使用时才开始的，主要是漏洞跟踪、监控扫描、安全突发事件的紧急响应。如表 6-2 所示介绍了安全测试与安全运维的区别。

表 6-2 安全测试与安全运维的区别

差异项	安全测试	安全运维
负责范围	业务系统安全	整个平台包括硬件、软件设备
介入时间	上线前	上线后
职责	漏洞扫描，渗透测试	日志分析，设施安全（硬件、防火墙、网络）
系统环境	测试环境	生产环境
目标	发现产品中存在的大部分安全缺陷（Bug）	建立安全防御体系，保障生产平台的稳定、持续运行，应急响应安全突发事件
团队合作方式	安全测试更多的工作集中在项目开发周期中，和整个项目团队紧密合作	运营的工作更多的集中在项目后期和生产运行时，和用户走得更近

6.1.4 安全测试工具

随着安全测试越来越广泛，业界出现了很多安全测试工具，覆盖信息收集、Web、数据库和操作系统等各个方面，有效地帮助用户进行安全测试。如表 6-3 所示是几款常用的安全测试工具。

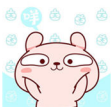


表 6-3 安全测试工具

序号	名称	内容
1	IBM AppScan	IBM AppScan 是一个领先的 Web 应用安全测试工具。它可以自动化进行 Web 应用的安全漏洞评估工作，能扫描和检测所有常见的 Web 应用安全漏洞，例如 SQL 注入（SQL-injection）、跨站点脚本攻击（cross-site scripting）、缓冲区溢出（buffer overflow）、最新的 Flash/Flex 应用及 Web 2.0 应用暴露等
2	Burp Suite	Burp Suite 是用于攻击 Web 应用程序的集成平台。它包含了大量的安全测试工具，并为这些工具设计了对外访问接口，以加快攻击应用程序的过程。它支持在整个测试过程中，分析应用攻击界面，结合手工和自动化技术，更快速、有效地找到安全漏洞
3	Metasploit	Metasploit 是一款开源的安全漏洞检测工具，可以帮助安全人员和 IT 专业人士识别安全性问题，挖掘漏洞，攻击漏洞，并评估漏洞风险级别
4	Wireshark	Wireshark 是一个适用于 Windows 和 Linux 的网络协议分析工具，也是一个很出名的数据包分析工具。它可以检查网络流量，是观察 TCP/IP 异常流量的最佳方法。此外，它也很适合分析其他安全工具的活动
5	Kail Linux	Kail Linux 是目前最流行的安全渗透测试平台，包含了最新的安全测试工具，它允许用户从 CD 或者 U 盘启动，通过 U 盘来实施安全渗透测试

6.1.5 安全测试用例

根据不同的安全测试类型，需要采用不同的测试方法来对测试项进行验证。如表 6-4 所示列出了常见的安全测试用例。

表 6-4 安全测试用例

测试类型	测试项	测试方法
配置管理	Tomcat 版本信息	查看 Tomcat 的版本，低版本的 Tomcat 存在一定的安全风险
	Spring ClassLoader 类远程代码执行	查看 Spring 框架版本，Spring 2.5.0~2.5.7 及 3.0.0~3.0.2 版本存在此漏洞
	不安全的 HTTP 方法	使用火狐插件 Live HTTP Headers 查看服务器所支持的 HTTP 方法
	目录遍历	手工测试，查看应用程序是否支持目录遍历模式
	Tomcat 管理后台访问控制	使用浏览器尝试访问 Tomcat 管理后台，并且查看后台是否存在弱口令
权限控制	任意文件下载	使用 WVS（Web Vulnerability Scanner，网络漏洞扫描）、截断工具进行测试，在文件下载点，通过控制路径参数进行测试
	纵向权限控制	使用 Burp Suite、Firebug 工具进行测试
	敏感文件访问控制	使用 wwwscan 等目录扫描工具进行扫描或手工检测



续表

测试类型	测试项	测试方法
身份认证与账户管理	路径遍历	使用 WVS、截断工具进行测试
	后台未授权访问	手工查看，不经过认证直接访问后台目录文件
	验证码机制缺陷	手工查看或使用 Burp Suite
	缺少账号锁定机制	手工测试，查看错误登录账号多少次后进行账号锁定
	账号和密码错误提示信息不相同	手工测试，查看系统对用户登录失败的提示信息
	用户超时机制	查看一段时间未操作是否自动销毁会话并且用户自动退出
	注销机制	查看注销后是否能够使用注销前的 Session
	密码修改机制	查看修改密码时是否验证原密码
	登录框 SQL 注入	使用 SQLMap 进行注入测试
	口令策略	手工注册系统用户，查看口令策略要求
数据存储与传输	明文传输	使用抓包工具抓包，如 HttpWatch 等，查看敏感数据（如账户口令等）是否明文进行传输
	明文存储敏感信息	查看数据库是否明文存储敏感信息（需考虑 SQL 注入漏洞）
会话管理	会话标识未更新	使用抓包工具抓包，如 Wireshark 等
	会话变量可猜测	查看会话变量标识是否可以猜测遍历
	未开启 cookie 中的 HttpOnly 属性	使用 Wireshark 或者 HttpWatch 进行抓包测试，查看系统是否启用 HttpOnly 属性
	加密算法	通过抓包工具获取加密数据，查看是否使用公开的强加密算法
	CSRF	使用 Burp Suite、Firebug 插件进行测试，检查系统是否对 Referer 进行验证，是否对表单增加随机 token
审核与日志	日志中记录了敏感信息	查看敏感信息是否支持 get 方式进行提交
	未开启相关日志审计功能	查看应用等日志是否开启
输入与输出	注入漏洞	使用 WVS 扫描器、火狐插件 Hackbar 进行测试
	文件上传	使用 Burp Suite 进行测试
	文件包含	使用 WVS、目录扫描器进行测试
	跨站攻击	使用 WVS、火狐插件 Hackbar、截断工具进行测试
	HTTP 消息头注入	使用 WVS 进行测试
	用户信息泄露	手工测试并使用抓包工具，查看用户敏感信息的展示是否进行了局部屏蔽
	基于参数的重定向钓鱼	手工测试，通过修改参数进行测试
	测试页面，默认页面	手工测试，上线前删除不需要的页面
	信息泄露	手工测试，查看系统在对用户敏感信息展现的过程中，是否进行了遮罩处理，不得全部暴露



续表

测试类型	测试项	测试方法
异常管理	缺乏异常管理机制	查看系统是否定制统一错误页面
逻辑漏洞	支付问题	使用 Burp Suite 进行测试
	认证问题	使用 Burp Suite 测试系统是否存在密码找回功能缺陷、非唯一用户名等问题
	修改提交表单	使用 Burp Suite、Firebug 等工具进行测试

6.2 开发安全规范

为了保证产品的安全性需求，降低安全漏洞修复的成本，在开发阶段就要制定和遵守开发安全规范，从源头上减少安全问题。常用的安全规范有：

- 跨站脚本安全规范；
- SQL 注入安全规范；
- 页面组件和敏感数据的安全规范；
- Java 安全规范；
- 应用集成安全规范。

6.2.1 跨站脚本安全规范

为避免与“层叠样式表（Cascading Style Sheets）”相混淆，故跨站脚本（Cross-site Scripting）简称为 XSS。XSS 是常见的 Web 攻击类型，攻击者利用应用程序的数据动态展示功能，在 Web 页面里嵌入恶意脚本代码。当用户浏览该页面时，嵌入在 Web 页面中的恶意代码会被执行，用户浏览器被攻击者挟持，从而达到攻击者的恶意目的。跨站脚本攻击有以下三种攻击形式。

- 反射型 XSS 或非持久 XSS

反射型 XSS 是指服务器直接从 HTTP 请求读取数据，并将其反射在 HTTP 响应中。当攻击者诱使用户向脆弱的 Web 应用程序提供恶意内容时，就会发生反射型 XSS 攻击，然后反射回受害者，并通过 Web 浏览器执行。提供恶意内容最常见的机制是将该内容作为 URL 的一个参数，该 URL 公开发送或直接通过邮件发送给用户。以这种方式构造的 URL 构成了众多钓鱼计划的核心，攻击者通过欺骗受害者访问脆弱网站，该网站将攻击者的内容反



射回受害者后，该内容就会在受害者的浏览器中执行。

● 存储型 XSS 或持续 XSS

存储型 XSS 是指应用程序在数据库、消息论坛、访客日志或其他可信的数据存储中存储危险数据。该危险数据随后被读回到应用程序，并被包含在动态内容中。从攻击者的角度看，注入恶意内容的最佳位置是展示给众多用户的区域或是展示给特定用户的区域。特定用户通常都具有应用程序中的较高权限，或能够接触到对攻击者有价值的敏感数据。如果这样的用户执行恶意内容，那么攻击者就代表该用户执行特权操作或者获取属于该用户的访问敏感数据的权力。例如，攻击者向日志消息中注入 XSS，管理员查看该日志的时候，并没有对其进行妥善处理，XSS 就可能被管理员执行。

● 基于 DOM 的 XSS

在基于 DOM 的 XSS 攻击中，客户端将 XSS 代码注入到页面中，而在反射型 XSS 和存储型 XSS 类型中是由服务器执行注入。基于 DOM 的 XSS 一般是发送给用户的、受服务器控制的、值得信赖的脚本，如 JavaScript，在用户提交表单之前，JavaScript 对该表单执行安全检查。如果服务器提供的脚本处理用户提供的数据，然后将数据注入到网页中（如动态 HTML），那么就可能发生基于 DOM 的 XSS 攻击。

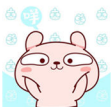
恶意攻击者利用跨站脚本攻击盗取用户 Cookie 等个人信息，伪造用户身份登录，向网站发送恶意请求，如果用户具有管理该网站的管理员权限，那么这种做法对于该网站尤其危险。攻击者可以控制用户浏览器，甚至接管用户的机器，结合浏览器及其插件漏洞，下载病毒木马到用户的计算机上执行；衍生 URL 跳转漏洞，模拟信任网站，使用钓鱼页面，诱骗用户输入密码，攻击者能够窃取用户在该网站上的账户信息。

为了防范跨站点脚本攻击，在产品开发时应遵循以下三个基本原则。

- 假定一切的输入和输出都是有害的，不要信任任何输入数据和输出数据。
- 所有传递过程都不能保证无侵入，所以在执行前、存储前和显示前都要进行“数据清洁”。
- 所有的数据校验、处理工作尽量在服务器端进行。

下面是对输入和输出参数的几种处理方式，可以最大限度地保持数据的清洁和安全。

(1) 处理 DOM 对象的输入参数。当操作页面中有 DOM 对象的时候，要对其输入的参数进行处理，防止 XSS 的注入，可使用 escape、encodeURIComponent、encodeURIComponent 方法



或自定义方法进行处理。类似代码如下：

```
Window.location=encodeURIComponent("http://www.securitytest.com/login?page=1");
```

部分需要进行编码处理的 DOM 对象操作如表 6-5 所示。

表 6-5 部分需要进行编码处理的 DOM 对象操作

document.location	Document.forms[...].action	window.execScript
document.location.hostname	Document.attachEvent	window.setInterval
document.location.replace	Document.create	window.setTimeout
document.location.assign	Document.execCommand	window.open
document.URL	Document.referrer	window.navigate
window.location.href	Document.write	

(2) 设置输入和输出字符校验的白名单和黑名单。只认为在白名单之内或者在黑名单之外的字符才是合法的。可以使用正则表达式对 “%” “?” “<” “>” “{” “}” “;” “&” “+” “=” “_” “.” “/” “\” “|” “#” “&” “|” “@” 等特殊符号进行过滤。

(3) 对输入和输出的特殊字符进行编码转义。将特殊字符转义成 HTML 实体编码 (ISO 8859-1 Latin1) 或其他编码，使得其在浏览器中不可自动执行。部分特殊字符的编码转义对照如表 6-6 所示。

表 6-6 部分特殊字符的编码转义对照

字符	转义字符	十进制	字符	转义字符	十进制
<	<	<	空格	 	
>	>	>	'	'	'
/	⁄	/	&	&	&
-	–	-	“	"	"

6.2.2 SQL 注入安全规范

SQL 注入 (SQL Injection) 攻击是黑客对数据库攻击的常用手段，主要是开发人员在编写代码时没有对用户输入的数据进行合法性判断，使应用程序存在安全隐患。当应用程序将用户输入的内容拼接到 SQL 语句中，一起提交给数据库执行时，就会产生 SQL 注入威胁。所以，SQL 注入攻击俗称为 “黑客的填空游戏”。



由于用户的输入也是 SQL 语句的一部分，所以攻击者利用这部分可以控制的内容，注入自己定义的语句，改变 SQL 语句的执行逻辑，让数据库执行自己需要的指令。通过控制部分 SQL 语句，攻击者可以查询数据库中任何自己需要的数据，利用数据库的一些特性，甚至可以获取数据库服务器的系统权限。

下面的代码是某个登录系统的 SQL 代码，通过输入用户名“username”和密码“password”，返回用户信息。

```
strSQL = "SELECT * FROM users WHERE (name = '" + userName + "')and (pw = '" + passWord + "');"
```

用户在用户名和密码中都输入“'1' OR '1'='1'”，将导致原本的 SQL 语句被填充为如下代码，作为无账号密码，亦可登录网站。

```
strSQL = "SELECT * FROM users WHERE (name = '1' OR '1'='1')and (pw = '1' OR '1'='1')";
```

随着大量 SQL 注入工具的出现，如 SQLInjection、SQLmap、Burp Suite 和 SQL Poizon 等，简化了 SQL 注入的过程，攻击者只需使用工具扫描一遍，就能达到攻击效果，增加了 SQL 注入的威胁。

防范 SQL 注入，需要坚持三个基本原则：

- 所有的用户输入都进行合法性校验；
- 所有的数据库 SQL 操作必须参数化；
- 数据库实现 Schema 分离，从设计角度上最大程度地减少 SQL 注入造成的危害。

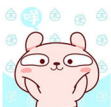
在实际开发过程中，遵守以下开发安全规范可以防止 SQL 注入。

(1) 尽量使用 PreparedStatement 代替 Statement。一方面，在大多数情况下，使用 PreparedStatement 的性能将优于使用 Statement 的性能；另一方面，可以最大限度地减少 SQL 注入发生的可能性。

(2) 用户提交的数据都应该进行合法性校验。禁止用户输入“'”“”“-”“%”“#”“&”“|”“@”“+”等有可能导致 SQL 注入的危险字符，避免给系统造成危害。

(3) 所有实现增加、删除、修改和查询操作的 DAO 必须继承一个公用处理类。

参数不能直接使用 String、StringBuffer 和 StringBuilder 组成 SQL 语句，而应该使用参数绑定的方式进行数据库查询和更新操作，使用占位符问号（“？”）解析 SQL 语句。



例如：

```
String userName = ctx.getAuthenticatedUserName();
String month = request.getParameter("dateTime");
String query =
    "SELECT * FROM T_LOGIN_LOG WHERE to_char(login_time,'yyyy-mm') = ? AND
owner=?";
Query stmt = sess.createQuery(query);
stmt.setString(0, month);
stmt.setString(1, userName);
List users= stmt.list();
```

(4) 对于敏感数据，限制客户端 IP 的返回数据量。比如，查询并修改登录用户的个人信息时，每次对同一个 IP 只返回一条记录，不能返回其他用户数据。

(5) 为当前应用建立权限比较小的数据库用户。不要使用 sa 这样的最高权限，避免数据库管理员丢失权限。

6.2.3 页面组件和敏感数据的安全规范

除了以上 XSS 防范和 SQL 注入防范，还要遵守一些基本的开发安全规范，对于页面组件和敏感数据处理需要参照以下几点。

1. 页面组件安全规范

在进行 Web 页面组件开发时，需要注意提交方式、编码方式和属性设置等。

(1) 页面标签必须关闭，属性值必须加引号。

在页面中，不关闭标签、不给属性值加引号，这些往往成为被攻击点，攻击者很容易利用这些漏洞进行注入。

(2) 表单提交方式必须使用 POST 方式。

表单默认的提交方式是 Get 方式，这种方式将表单中的数据按照“variable=value”的形式，使用“?”添加至 Action 所指向的 URL 后面，各个变量之间使用“&”连接。所要传递的信息除了受 URL 长度限制之外，内容都暴露无疑。而表单提交使用 POST 方式，就可以保护数据不被泄露，其代码如下：

```
<form action="..." method="post" target="_blank">
...
```




```
</form>
```

(3) 所有的非 ASCII 字符在 URL 中传递时都需要按照协商好的编码方式使用 URL 编码。

推荐使用 JavaScript 中的 `encodeURIComponent` 方法实现。`encodeURIComponent` 默认都返回 UTF8 编码的 URL，区别在于 `encodeURIComponent` 方法不会对字符 “:” “/” “;” 和 “?” 进行编码，而 `encodeURIComponent` 则会对这些字符进行编码处理。

(4) 尽量使用页面对象的 `innerText`，不要使用 `innerHTML` 属性。

对象的 `innerText` 属性默认会对输入的数据进行编码转换。数据中包含有 HTML 的可执行标签（如 `<Script>`）时不会直接被执行，而使用 `innerHTML` 属性可能会引入恶意脚本，恶意代码注入实例如下：

```
<SPAN id="Addr "></SPAN>
<Script>
// Using innerText renders the content safe.
Addr.innerText="... ";
//Using innerHtml requires the use of HtmlEncode to make it safe.
Addr.innerHTML="...<Script>...</Script> ";
</Script>
```

(5) 输入框最大长度限制和输入数据类型限制。

输入框一般是攻击者重点的攻击对象，攻击者可以通过输入框进行数据窃取（如 SQL 注入）或者制造危害（如 XSS）。开发人员通过对输入框的最大长度和输入数据类型进行限制，可以从一定程度上防范此类攻击。

(6) 关闭客户端自动完成功能，减少驻留在客户端的数据。

设置 `AUTOCOMPLETE` 属性为 `OFF`，关闭客户端的自动完成功能，减少驻留在客户端的数据。

```
<FORM ... AUTOCOMPLETE="OFF"/>
<INPUT ... AUTOCOMPLETE="OFF"/>
```

(7) 设置 `<Frame>`/`<iFrame>` 受限，防止脚本执行。

将 `<Frame>`/`<iFrame>` 中的 `security` 属性设置为 `restricted` 后，`Frame` 中的脚本将不能执行（仅限于 IE）。实例如下：



```
<iFrame security="restricted" src="http://www.securitytest.com/login.htm" />
```

(8) 设置 URL、图片等资源访问的白名单。

2. 敏感数据的安全规范

敏感数据包括 Cookie、Session 和账号密码等，这些数据的泄露会造成严重的后果，因此在开发时，尤其要注意，一定要遵守以下原则。

- 不能随意在 Cookie、Session 和 ServletContext 中存放数据，在这些对象中存放的数据必须有统一的定义说明和生命周期的管理等。
- 敏感信息要保证其私密性。

在页面显示、操作交互中不可避免地会有用户的标识信息、账号、密码、金额信息等敏感数据（保密性的数据）的存在。为保证这些数据不被泄露，开发时必须对所有敏感信息进行加密处理，不能以明文的形式存在于任何网络、内存及其他持久化介质中（如数据库、文件磁盘系统等）。

在开发过程中，要遵守以下几条细则。

(1) 所有的密码、账号等敏感信息都不能在 URL 中明文传递。

(2) 所有的密码、银行账号等敏感信息都不能存储到 Cookie、Session 和 ServletContext 中。

(3) 防止信息泄露。

- 在系统上线时，使用的 log level 对应的日志输出中不允许包含任何密码和账号等机密信息。
- SVN 或 GIT 集成分支上的代码中不允许存留可用的 system.out 或者 err.print 语句。
- 在测试和调试阶段所使用的测试页面、单元测试模块等不允许提交到 SVN 或 GIT 集成分支上。
- 不允许将系统产生的错误异常信息直接展示给用户，需要有统一的错误处理。

(4) 尽量减少不必要的信息传递。

在信息的传递过程中，如果目前步骤中的对象、对象属性和参数等在下一个步骤中不需要，则不要再进行传递，甚至可以显式地销毁，这样可以有效地减小信息被截获的几率，特别是敏感数据（如用户密码、账户信息等）。

6.2.4 Java 安全规范

在使用 Java 开发软件的过程中，需要特别注意以下几点。

(1) Cookie 的安全防范。

- 存储于 Cookie 中的敏感数据必须加密。
- 没有特殊要求时，尽量使用会话 Cookie（非持久化）。
- 如果使用持久化 Cookie，应该设置 Cookie 超时。
- 激活 Cookie 安全传输，表示创建的 Cookie 只能在 HTTPS 连接中被浏览器传递到服务器端进行会话验证，如果是 HTTP 连接则不会传递该信息。

Java 代码实例如下：

```
Cookie[] cookies = request.getCookies();
for(Cookie cookie : cookies){
    if(...){
        // 设置 Cookie 非持久化到客户端磁盘上
        cookie.setMaxAge(-1);
        // 设置 Cookie 超时, 120 秒
        cookie.setMaxAge(120);
        // 激活 Cookie 安全传输
        cookie.setSecure(true);
        response.addCookie(cookie);
    }
}
```

(2) 必须设置 Session 的超时时间。

在 web.xml 中设置 session-timeout，单位是秒。代码实例如下，其中设置 Session 超时为 10 秒。

```
<session-config>
    <session-timeout>10</session-timeout>
</session-config>
```

(3) 必须构建统一的错误处理页面。

在 web.xml 中设置 error-page，根据 error-code 导向对应的错误页面。代码实例如下，在 HTTP 页面返回 error code 为 500 时，显示页面 error.jsp。

```
<error-page>
```



```
<error-code>500</error-code>
<location>/error.jsp</location>
</error-page>
```

(4) 多线程中线程安全的防范。

- 尽量少用静态 (static) 变量和静态方法 (除了静态常量 static final constants)。
- 尽量使用多线程框架 (java.util.concurrent)，构建多线程同步机制。
- 使用 ThreadLocal 类，避免多个线程之间类成员的共享冲突。
- 如非必要，不要使用 synchronized 关键字，使用不当会造成死锁。必须要使用 synchronized 关键字时，应将同步范围最小化，即将同步作用域用到最需要的地方，避免大量的同步块或同步方法等。

(5) 增加 Referer 的检查，防止非法访问。

Referer 是 HTTP Header 中的一个字段，当浏览器向服务器发送请求时，Referer 通知服务器请求发起的位置。使用如下 Java 代码，对请求来源进行验证和过滤。

```
If(!request.getHeader("REFERER").equals("http://www.securitytest.com/index.html")){
Handling...
}
```

(6) 构建页面流的访问控制，建议使用 Spring Web Flow。

6.2.5 应用集成安全规范

各个应用之间不应有内在的信任关系，应用相互访问时，被访问的系统要对访问的系统进行认证和授权验证。

6.3 代码安全审核工具 Checkmarx

Checkmarx 是著名的源代码安全扫描软件，可用于识别、跟踪和修复源码中的技术和逻辑缺陷，也能提前发现安全漏洞和质量缺陷等问题。它通过虚拟编译器对源码进行自动分析或者设定自动扫描时间，定期收到扫描结果。它的扫描结果显示为静态报表形式，能对扫描结果进行审查，消除误报。

Checkmarx 支持 20 多种语言的源码扫描，包括最新的开发框架，却不需要任何代码构建或编译，仅提供源码即可，不占用开发人员的正常开发时间。

下面是 Checkmarx 汇报的几个安全漏洞实例。

6.3.1 SQL 注入

SQL 注入的风险等级为高风险。Checkmarx 扫描出的 Java 源代码摘录如图 6-3 所示。

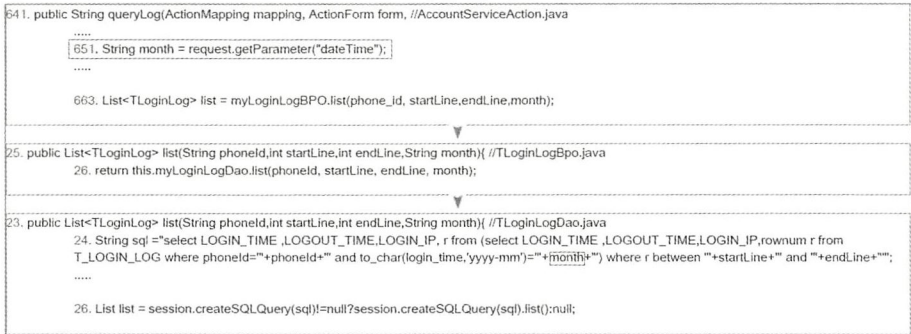


图 6-3 SQL 注入代码

AccountServiceAction.java 文件中的第 651 行接收了从 request 传过来的变量“dateTime”，并赋予给变量 month，该变量为用户自定义输入。在 TLoginLogDao.java 文件的第 24 行中，在没有经过任何过滤的情况下，拼接到 SQL 语句中。恶意用户可以输入任意的 SQL 恶意字符来控制 SQL 语句的行为，造成了 SQL 注入漏洞。

以下为修复建议：

(1) 对于 SQL 注入的预防，最好的方式是使用带参数化的动态 SQL 查询，具体参考 6.2.2 节。

(2) 有时也可以使用黑名单和白名单对输入字符做处理，例如只允许字母、数字和空格就可以修复该问题。代码实例如下：

```
private String clean(String s)
{
    StringBuffer clean = new StringBuffer();
    for (int loop = 0; loop < s.length(); loop++)
    {
        char c = s.charAt(loop);
```

```
//判断字符是否为字母、数字或者空格
if (Character.isLetterOrDigit(c) || Character.isWhitespace(c))
{
    clean.append(c);
}
else
{
    clean.append('.');
}
}
```

6.3.2 反射型跨站脚本攻击

反射型 XSS 发生在 Web 客户端提交未经验证的用户数据到 Server 端时, 用户数据没有被编码转义, 从而造成可执行代码被嵌入到动态页面中。

反射型 XSS 的风险等级为高风险。Checkmarx 扫描出的反射型跨站脚本攻击代码摘录如图 6-4 所示。

```
85. private void printJsonPOut(HttpServletRequest request,PrintWriter out,Object obj){ //GetUserInfoAction.java
.....
87. String call = request.getParameter("jsoncallback");

88. StringBuffer sb = new StringBuffer(call);
.....

93. out.print(sb.toString());
```

图 6-4 反射型跨站脚本攻击代码摘录

在 GetUserInfoAction.java 文件的第 87 行中, “jsoncallback” 参数是用户从客户端输入并传过来的数据, 在第 93 行中使用 out.print 语句直接输出到了页面。如果用户在 Web 页面按正常的方式输入, 则程序不会出现任何问题, 但是如果用户在输入框中输入 Java Script、Flash 链接及其他任何 Web 浏览器可以执行的脚本, 那么就会导致跨站脚本攻击。这种攻击会导致被攻击者的 Cookie 或者 Session 等私有信息泄露, 攻击者也可以操控浏览器或在用户的电脑上注入木马等恶意程序。

任何从数据库或文件系统中得到的数据都应当视为不安全的, 都有可能导致跨站脚本攻击的发生。在把未经验证的数据传送给 Web 页面处理前, 要对数据进行清洁处理, 可以使用输入验证的白名单或者黑名单的方法, 也可以使用成熟的输出 encoding 方法把一些



Web 浏览器易解析成命令的特殊字符的 ASCII 码转换成 HTML 的编码，参见表 6-6 规范中列举的编码转义。以下是代码实例：

```
public String encodeHtml(String input)
{
    StringBuffer out = new StringBuffer();
    for (int i = 0; i < input.length(); i++)
    {
        char c = input.charAt(i);
        if (c == '<')
        {
            out.append("&lt;");
        }
        else if (c == '>')
        {
            out.append("&gt;");
        }
        else if (c == "\"")
        {
            out.append("&quot;");
        }
        else if (c == "\'")
        {
            out.append("&apos;");
        }
        else if (c == '&')
        {
            out.append("&amp;");
        }
        else if (c > 0x20 && c < 0x126)
        {
            out.append(c);
        }
        else
        {
            out.append("&#" + (int)c + ";");
        }
    }

    return out.toString();
}
```




```
}
```

同时也可以借用一些已有的函数在输出之前对字符串进行处理，如 `HtmlEncoder.encode`、`ServletResponse.encode` 或者自定义验证处理。

6.3.3 储存型 XSS

动态生成的 Web 页面接受未经验证或未经过滤的用户输入，从而允许攻击者在网页中嵌入恶意脚本代码。任何人访问该网页时，服务器上的这些脚本代码都会被执行，从而危及客户端的安全，会造成储存型 XSS 漏洞。

储存型 XSS 的风险等级为高风险。Checkmarx 扫描出的储存型 XSS 代码摘录如图 6-5 所示。

```
31. ResponseDTO socketSend(RequestDTO dto){  
  
    //MailUtil.java  
    .....  
    47. String rin=line.readLine();  
    .....  
  
    49. out.println(rin);  
}
```

图 6-5 储存型 XSS 代码摘录

在 `MailUtil.java` 文件中的第 47 行，从文件读取一行数据，并在第 49 行打印输出。而在这个过程中并未对数据进行输入验证。如果数据中包含可被执行的恶意脚本，那么浏览器就会去执行这些恶意的脚本，导致储存型 XSS 攻击的发生。

修复建议类似于反射型 XSS，也使用白名单和黑名单验证，对于输入的数据进行编码转义。

6.3.4 HTTP 响应头分裂（Http_Response_Splitting）

将未经验证的数据写入 HTTP Header 会使攻击者窃取服务器端返回的所有 HTTP Response。当 HTTP request 中包含意外 CR 和 LF 字符时，就会出现 HTTP 响应头分裂的情况。服务器可能用被解释为两种不同的 HTTP 响应输出流（而不是一个）来进行回应。攻击者可以控制第二个回应，从而发动攻击，如跨站点脚本攻击和缓存中毒攻击。



HTTP 响应头分裂的风险等级为中等风险。Checkmarx 扫描出的 HTTP 响应头分裂代码摘录如图 6-6 所示。

```
429. public ActionForward toQueryFeeServiceByMonth(ActionMapping mapping, ActionForm form, //BusinessDisplayAction.java
.....
432. String businessId=request.getParameter("businessId");
.....
442. response.sendRedirect(request.getContextPath()+"/business/queryFeeServiceBusinesses.do?businessId="+businessId);
```

图 6-6 HTTP 响应头分裂代码摘录

在 BusinessDisplayAction.java 文件中的第 432 行，接收了从 request 传过来的变量“businessid”，该变量为用户输入。在程序的第 442 行，没有经过任何验证就把用户输入设置到 response 的 redirect 中。恶意攻击者可以通过输入 CR 和 LF 字符来拆分 HTTP 响应头，从而制造出多个 HTTP 响应输出流。

为修复此类问题，要非常谨慎地构筑 HTTP Header，避免使用未经验证的输入数据。应假设所有输入数据都是恶意的。使用黑名单和白名单的合适组合来确保系统只对有效的和预计的输入数据进行处理。输入 HTTP Header 数据时应该过滤 CR 和 LF 等特殊字符，实例代码如下：

```
public static String replaceBlank(String str) {
    String dest = "";
    if (str!=null) {
        Pattern p = Pattern.compile("\\s*|t|r|n");
        Matcher m = p.matcher(str);
        dest = m.replaceAll("");
    }
    return dest;
}
```

6.4 安全漏洞

6.4.1 信息收集

信息收集是渗透测试的基础，在制定渗透测试策略之前，通过主机探测手段收集主机信息，例如服务器的版本、开放的服务和端口。通过收集的信息制定攻击路线，为后期的



渗透测试提供参考。

1. 主机探测

在信息收集, 主要使用 Nmap 工具来收集主机信息。使用 Nmap 对 10.100.1.65 这台服务器进行扫描探测, 扫描结果如图 6-7 所示。

```
msf auxiliary(arp_sweep) > nmap -A 10.100.1.65
[*] exec: nmap -A 10.100.1.65

Starting Nmap 5.51SVN ( http://nmap.org ) at 2015-12-10 16:21 CST
Nmap scan report for 10.100.1.65
Host is up (0.00023s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.2.2
|_ ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ drwxr-xr-x  2 0      0      4096 Jul 24 00:49 pub
22/tcp    open  ssh      OpenSSH 5.3 (protocol 2.0)
|_ ssh-hostkey: 1024 e4:d5:78:c8:e8:71:bb:37:3f:d9:92:f2:43:0e:ea:21 (DSA)
|_ 2048 26:cc:b5:53:ee:19:34:3d:24:5b:c5:11:28:2a:27:ce (RSA)
8099/tcp  open  ajp13    Apache/Jserv-(Protocol-v1.3)
8080/tcp  open  http     Apache Tomcat/Coyote JSP engine 1.1
|_ http-title: Apache Tomcat/7.0.64
|_ http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_ http-favicon: Apache Tomcat
|_ http-open-proxy: Proxy might be redirecting requests
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=5.51SVN/D=12/10/OT=21/CT=1%CU=43981%PV=Y%DS=2%DC=T%G=Y%TM=5669362
OS:5%P=1686-pc-linux-gnu)SEQ(SP=105%GCD=1%ISR=109%TI=Z%CI=Z%II=1%TS=A)OPS(O
OS:I=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NT11NW7%O4=M5B4ST11NW7%O5=M5B4ST11N
OS:W7%O6=M5B4ST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3890)ECN(R
OS:Y%DF=Y%T=40%W=3908%O=M5B4NNSMW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=0%A=S%F=AS%
OS:RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%F=0%Q=)T5(R=Y
OS:RDF=Y%T=40%W=0%S=Z%A=S%F=AR%O=RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%F=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S%F=AR%O=RD=0%Q=)U1(R=Y%DF=N%T=40%CD=S
OS:5)

Network Distance: 2 hops
Service Info: OS: Unix
```

开放 FTP 服务, 允许匿名登录, 能查看到 pub 文件的权限

查看到服务器版本信息

查看到主机信息

图 6-7 主机探测结果

在本实例中, 可以看出一些关键的服务器信息:

- 服务器开放了 FTP、SSH 服务, 并允许匿名登录;
- 可以查看 FTP 目录的 pub 文件夹;
- 服务器操作系统是 UNIX;
- 服务器安装了 Apache Tomcat 7.0.64。

以上信息说明这台服务器是一个 Web 服务器, 可能运行着一些 Web 应用程序, 可以进一步对这台服务器进行 Web 攻击和 FTP 攻击, 获取更多系统安全漏洞。

2. 探测结果

对前面收集的信息进行汇总, 汇总信息如表 6-7 所示。



表 6-7 探测信息汇总

主机	操作系统	开放的端口	对应的服务器版本
网站服务器 (10.100.1.65)	Linux/UNIX	SSH (22)	OpenSSH 5.3 (Protocol 2.0)
		HTTP (8080)	Apache Tomcat 7.0.64
		AJP (8009)	Apache JServ (Protocol Version 1.3)
		FTP (21)	Vsftpd 2.2.2

把获取的端口和服务信息进行汇总，并按照可能的攻击路径对其进行分类。

- 这台网站服务器由于开放了 SSH 和 FTP 服务，所以可以对这两个服务的用户和口令进行远程拆解攻击。
- 可以利用漏洞扫描技术发现其中存在的安全漏洞，然后对这些服务进行漏洞渗透利用。
- 服务器开放了 Apache 的 8009 和 8080 端口，可以发现在 Apache Tomcat 上运行着一系列的 Web 应用程序，对此可以实施 Web 应用漏洞扫描探测与渗透攻击。

3. 攻击路线

根据主机探测和探测结果分析，可以确定如表 6-8 所示的攻击路线，进行后期的渗透测试。

表 6-8 攻击路线

可能的攻击路线	攻击对象
口令猜测嗅探	10.100.1.65: SSH、FTP
系统漏洞深入扫描利用	使用 Metasploit 等渗透工具，更深层次地挖掘系统存在的安全漏洞
Web 应用漏洞扫描利用	10.100.1.65: Tomcat

6.4.2 口令入侵

口令入侵是指使用某些合法的账号和口令登录到目标主机，然后实施攻击活动。账号和口令可以通过以下两种途径获取：

- 网络监听；
- 口令猜测—暴力破解。

1. 网络监听

通过网络监听非法获得用户口令，这类方法有一定的局限性，但危害性极大。监听者往往采用中途截击的方法，这也是获取用户账号和密码的一条有效途径。目前，很多协议



并没有采用任何加密或身份认证技术，如在 Telnet、FTP、HTTP、SMTP 等传输协议中，用户账号和密码信息都是以明文格式传输的，此时若攻击者利用数据包截取工具便可以很容易地收集到用户的账号和密码。

还有一种中途截击的攻击方法，它同服务器端完成“三次握手”并建立连接之后，在通信过程中扮演“第三者”的角色，假冒服务器身份欺骗用户，再假冒用户向服务器发出恶意请求，其造成的后果不堪设想。

另外，攻击者有时还会利用软件和硬件工具时刻监视系统主机的工作，等待并记录用户登录信息，从而取得用户密码；或者编写有缓冲区溢出错错误的 SUID 程序来获得超级用户权限。比较著名的工具有 Wireshark，通过它可以监听网络中传输的数据，探测用户名和密码，如图 6-8 所示。

No.	Time	Source	Destination	Protocol	Length	Info
32	9.000470000	10.108.5.208	10.108.12.160	TCP	54	58131 > http-alt [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
35	5.060145000	10.108.5.208	10.108.12.160	TCP	54	58131 > http-alt [ACK] Seq=2 Ack=2 Win=256 Len=0
47	6.061099000	10.108.5.208	10.108.12.160	TCP	54	58126 > http-alt [ACK] Seq=1 Ack=2 Win=256 Len=0
87	12.330704000	10.108.5.208	10.108.12.160	TCP	54	58125 > http-alt [FIN, ACK] Seq=1 Ack=2 Win=256 Len=0
89	13.232183000	10.108.5.208	10.108.12.160	TCP	66	58169 > http-alt [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=256 SACK_PERM=1
91	12.332552000	10.108.5.208	10.108.12.160	TCP	54	58169 > http-alt [ACK] Seq=1 Ack=1 Win=65536 Len=0
92	12.336880000	10.108.5.208	10.108.12.160	HTTP	938	POST /jdeploy/login HTTP/1.1 (application/x-www-form-urlencoded)
95	12.350729000	10.108.5.208	10.108.12.160	HTTP	821	GET /jdeploy/login.jsp?error=true HTTP/1.1
98	12.354519000	10.108.5.208	10.108.12.160	TCP	54	58169 > http-alt [ACK] Seq=1652 Ack=2140 Win=65536 Len=0
107	12.557320000	10.108.5.208	10.108.12.160	TCP	66	58169 > http-alt [ACK] Seq=1652 Ack=3056 Win=64768 Len=0 SLE=2140 SRE=3056
339	32.373805000	10.108.5.208	10.108.12.160	TCP	54	58169 > http-alt [ACK] Seq=1652 Ack=3057 Win=64768 Len=0
446	35.058390000	10.108.5.208	10.108.12.160	TCP	54	58169 > http-alt [FIN, ACK] Seq=1652 Ack=3057 Win=64768 Len=0
460	35.172480000	10.108.5.208	10.108.12.160	TCP	66	58169 > http-alt [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=256 SACK_PERM=1
464	36.177785000	10.108.5.208	10.108.12.160	TCP	54	58196 > http-alt [ACK] Seq=1 Ack=1 Win=65536 Len=0
466	36.184390000	10.108.5.208	10.108.12.160	HTTP	943	POST /jdeploy/login HTTP/1.1 (application/x-www-form-urlencoded)
469	36.196050000	10.108.5.208	10.108.12.160	HTTP	832	GET /jdeploy/login.jsp?error=true HTTP/1.1
472	36.198489000	10.108.5.208	10.108.12.160	TCP	54	58196 > http-alt [ACK] Seq=1674 Ack=2140 Win=65536 Len=0

Frame 466: 949 bytes on wire (7592 bits), 949 bytes captured (7592 bits) on interface 0	
Interface id: 0	
Encapsulation type: Ethernet (1)	
Arrival time: Apr 11, 2016 10:55:42.199418000	
[Time shift for this packet: 0.000000000 seconds]	
Epoch Time: 1460343342.199418000 seconds	
[Time delta from previous captured frame: 0.005170000 seconds]	
[Time delta from previous displayed frame: 0.010813000 seconds]	
[Time since reference or first frame: 36.188598000 seconds]	
Frame number: 466	
Frame length: 949 bytes (7592 bits)	
Capture length: 949 bytes (7592 bits)	
[Frame is marked: False]	
[Frame is ignored: False]	

0230	66	66	61	74	65	0d	0a	41	63	63	65	70	74	2d	4c	61	flate..A ccept-La
0240	66	67	75	61	67	65	3a	20	7a	68	2d	43	4e	2c	7a	68	nguage: zh-CN,zh
0250	3b	73	3d	30	2e	38	0d	0a	43	6f	6f	6b	69	65	3a	20	;q=0.8. Cookie:
0260	44	53	45	32	33	39	46	35	43	44	3d	37	43	33	33	42	355550N ID=C33b
0270	34	45	42	33	39	46	35	32	34	38	39	36	36	39	38		46E239P5 20896698
0280	32	46	44	30	37	34	32	37	38	32	36	3b	20	48	6d	5f	2FD07427 826; HL
0290	66	76	74	5f	63	34	35	37	38	62	64	61	39	30	34		Vvt_caf57 R8n600D4
02a0	30	65	63	31	39	64	65	64	35	36	31	31	32	62	38	32	0ec19ded 56112b82
02b0	32	34	36	66	6d	31	34	35	33	39	36	33	35	35	35	2c	246f145 3086555
02c0	31	34	33	33	38	38	38	35	35	3c	31	34	33	33	33		14510868 55 145131
02d0	36	36	31	31	30	2c	21	34	35	35	36	37	30	34	37	34	66110.14 55670474
02e0	30	20	5f	5f	74	6d	61	3d	32	36	38	35	37	39			utma=2618579
02f0	33	33	2e	23	31	35	38	30	36	33	34	39	2e	31	34	35	33.31580 6349.145
0300	32	30	36	31	10	55	32	2e	31	34	34	34	36	37	30		2061052.14544670
0310	33	34	2e	31	34	33	39	32	32	30	38	34	35	2e	33	30	34.14592 20845.31
0320	20	5f	5f	75	74	6d	74	3d	32	36	31	38	35	37	39	33	utnz=26185793
0330	2e	21	34	35	32	30	36	31	30	35	32	2e	31	2e	31		3145206 1052.1.1
0340	2e	75	74	6d	63	73	72	3d	28	64	69	72	65	63	74	29	.utmcsr=(direct)
0350	7c	75	74	6d	63	6e	3d	28	64	69	72	65	63	74	29		utmccm=(direct)
0360	7c	75	74	6d	63	6d	3d	28	64	6f	6e	65	59	3b	20		utmcmd=(none)
0370	5f	5f	75	74	6d	76	3d	32	36	31	38	35	37	39	33	33	utmw=2 81857933
0380	2e	75	74	6d	72	65	61	3d	28	64	69	72	65	63	74	29	1174744 time=0
0390	31	0d	0a	0d	0a	75	73	65	72	6e	61	6d	65	3d	61	64	1...use rname=ad
03a0	6d	69	6e	6d	6e	6d	61	73	73	77	6f	72	64	3d	61	64	ntid=ass word=adm
03b0	6d	69	6e	6d	6e	6d	61	73	73	77	6f	72	64	3d	61	64	int32

图 6-8 使用 Wireshark 监听用户名

2. 口令猜测

口令猜测攻击是攻击者根据口令规则编写口令字典库，然后依据该字典库通过猜测获得用户的口令。通常，攻击者根据口令的长度、口令的特征和各种字符组合方式选择某个口令进行攻击注入。口令猜测流程如图 6-9 所示。



质量全面管控：从项目管理到容灾测试

(1) 攻击者选择一个网络账号，比如有些网站用户信息泄露，就使得攻击者比较容易获得账号。

(2) 从弱口令字典库中选取一个弱口令，然后进行口令验证。

(3) 如果口令验证通过，那么口令猜测成功，流程结束。

(4) 如果口令验证不通过，那么继续遍历字典库。字典库所有口令都遍历一遍，那么意味着口令猜测失败，流程结束。后续可以更新口令字典库，然后重新开始新的口令猜测流程。

(5) 如果遍历字典库后，仍有剩余弱口令，那么重新获得一个弱口令，从第(2)步开始继续口令猜测。

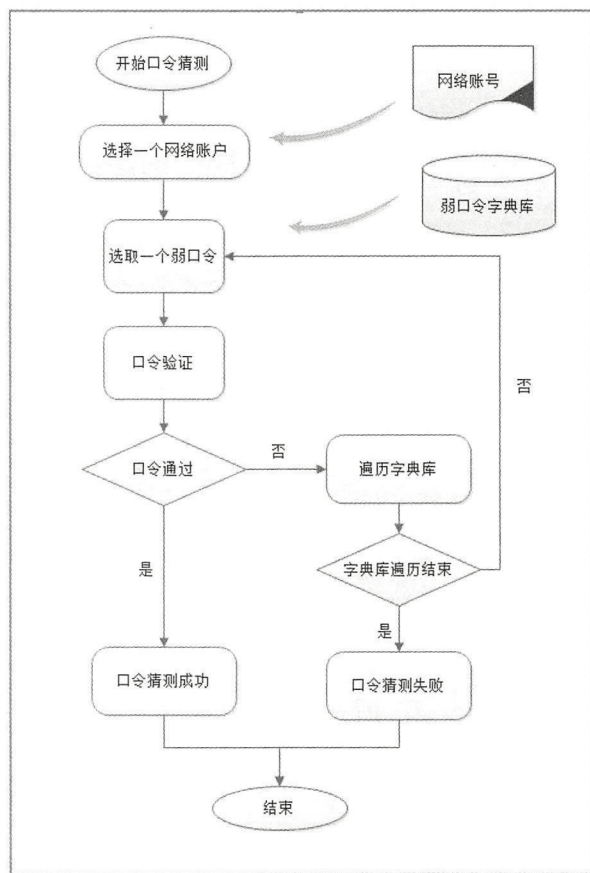


图 6-9 口令猜测流程

不同的口令类型有不同的破解工具，例如 SMB 服务器、数据库、POP3 都有对应的破



解工具。常见的破解工具如下所示：

- Web 口令破解

Burp Suite 工具提供口令暴力破解功能，有默认的口令字典，也允许自定义口令字典。在访问 Web 应用时，可重复发送登录请求以获取正确的账号和密码。

- SMB 口令破解

NAT (NetBIOS Auditing Tool) 是 SMB 服务器口令破解工具。

- Telnet 口令破解

Brutus 是远程密码破解工具，支持 Windows 操作系统，可以破解 HTTP、POP3、FTP、SMB 和 Telnet 等密码。

- 数据库口令破解

SQLdict 是远程破解数据库口令工具，支持 Windows 操作系统。

- POP 3 口令破解

EmailCrack 是一个基于 POP3 协议的口令破解软件，它根据攻击者提供的用户名单和口令列表文件，逐个猜测用户口令。

6.4.3 心脏滴血漏洞

在网络安全界，对网络安全影响最大的是 OpenSSL 的“Heartbleed”安全漏洞，在我国被称为“OpenSSL 心脏滴血漏洞”、“击穿心脏”等。2014 年 4 月 7 日，来自 Codenomicon 公司和谷歌安全部门的研究人员发现 OpenSSL 的源代码中存在一个漏洞，使得黑客可以窃取 OpenSSL 服务器内存中的私用密钥。由于密钥的重要性，从这个意义上讲，该漏洞是互联网安全史上最严重的漏洞之一。

OpenSSL 心脏滴血漏洞的原理是，OpenSSL 引入了心跳 (heartbeat) 机制来维持 TLS 连接，心跳机制是作为 TLS 的扩展实现，代码中包括 TLS (TCP) 和 DTLS (UDP) 都没有做边界检测，导致攻击者可以利用这个漏洞来获得服务器内存中的一些数据，每次至少可以获得 16KB 数据，从理论上讲最多可以获取 64KB 数据。用户利用 TLS 连接，发起 Client Hello 询问，测试服务器是否正常在线，服务器发回 Server Hello 响应，表明正常建立 SSL 通信。每次询问都会附加一个询问的字符长度 pad length，此时若 pad length 大于实际的长度，则服务器返回和 pad length 同样规模的字符信息，于是造成了内存中信息的越界访问。

一位安全行业人士曾透露利用这个漏洞在某著名电商网站上尝试读取数据，在读取 200



质量全面管控：从项目管理到容灾测试

次之后，获得 40 多个用户名和 7 个密码，利用这些用户名和密码，成功地登录了该电商网站。GitHub 上的一个统计数据分析了 2014 年 4 月 8 日前 1000 个访问量最大的网站，发现受影响的网站有 Yahoo、Imgur、Stack Overflow、Slate 和 DuckDuckGo。安全的网站有 Google、Facebook、Wikipedia、Twitter 和 Amazon。

1. 攻击脚本

常见的攻击方式如下。

- Nmap 脚本 ssl-heartbleed.nse。

脚本下载地址：<https://nmap.org/nsedoc/scripts/ssl-heartbleed.html>。

- Jared Stafford 的 testssl.py。

脚本下载地址：<https://gist.github.com/sh1n0b1/10100394>。

- CSHearbleedScanner。

工具下载地址：<http://www.crowdstrike.com/community-tools/>。

2. 实例

使用 OpenSSL 漏洞检测工具进行检查，如图 6-10 所示是心脏滴血漏洞的一个实例。使用方法如下：

```
python openssl.py <ip> -p <port>
```

在本例中，使用 Python 脚本 openssl.py 对服务器的端口 8443 发起 TLS 连接，服务器返回内存中的信息。如图 6-10 所示方框处的 Received heartbeat response 就是返回的内存数据。

```
d:>python openssl.py 196.66.50.03 -p 8443 ! more
Connecting...
Sending Client Hello...
Waiting for Server Hello...
... received message: type = 22, ver = 0302, length = 66
... received message: type = 22, ver = 0302, length = 3080
... received message: type = 22, ver = 0302, length = 331
... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
... received message: type = 24, ver = 0302, length = 16384
Received heartbeat response:
0000: 02 40 00 D8 03 02 53 43 5B 90 9D 9B 72 0B BC 0C .e...SC[...P...
0010: BC 2B 92 A8 48 97 CF BD 39 04 CC 16 0A 85 03 90 .+.H...9.....
0020: 9F 77 04 33 D4 DE 00 00 66 C0 14 C0 0A C0 22 C0 .v.3...f....."
0030: 21 00 39 00 38 00 88 00 87 C0 0F C0 05 00 35 00 !.9.8...../5.
0040: 84 C0 12 C0 08 C0 1C C0 1B 00 16 00 13 C0 0D C0 .....
0050: 03 00 0A C0 13 C0 09 C0 1F C0 1E 00 33 00 32 00 .....3.2.
```

图 6-10 心脏滴血漏洞实例



3. 解决方案

如何有效地修补心脏滴血漏洞？最简单、有效的修补方案是升级到 OpenSSL 1.0.1g 以上，或者使用“-DOPENSSL_NO_HEARTBEATS”选项重新编译 OpenSSL，从而禁用易受攻击的功能，直至可以更新服务器软件。目前 Snort 已经列出相关规则，可以用于 IDS/IPS 拦截，也有很多第三方 CDN 流量规则拦截。

用户在登录网站时如何确保自己的账户和密码是安全的呢？所幸现在出现了很多检测该漏洞的服务和工具，若是不安全的网站，那么一定要避免使用任何涉及账号和密码的交易功能。下面列举几个常用的检测工具。

- Heartbleed test

该检测工具是由意大利安全专家 Filippo Valsorda 率先开发的，模拟 OpenSSL Heartbleed 漏洞的入侵方法，仅需输入网址，便可以测试是否受 Heartbleed 漏洞的影响。

- possible.lv Heartbleed test

它是由安全公司 Possible.lv 开发的 Heartbleed 检测工具。

- LastPass Heartbleed Checker

它是由密码管理工具厂商 LastPass 开发的检测工具。

- Qualys SSL Labs Server Test

由 Qualys 公司 SSL 实验室提供的 SSL 测试，测试服务器是否受 Heartbleed 漏洞影响，还可以评测服务器的加密安全性。

- Chrombleed

Chromebleed 是 Google Chrome 浏览器的扩充功能，每次浏览网页时，便会主动弹出网页是否受 Heartbleed 漏洞影响的提示信息，保证访问网站的用户名和密码安全。

6.5 OSSIM 安全管理平台

随着互联网的发展，伴随而来的网络安全问题日益严重。网络安全威胁的复杂程度越来越高，不再局限于传统或单一的病毒或攻击。网络攻击更多是融合病毒、木马、间谍、扫描技术于一身的混合式攻击。黑客利用蠕虫制造僵尸网络，整合更多的攻击源，对目标集中展开猛烈的拒绝服务攻击。面对这种情况，需要把防火墙、防病毒系统、入侵检测系统、漏洞扫描系统等进行整合，在资源共享的基础上建立一个集中式平台，例如 OSSIM，

质量全面管控：从项目管理到容灾测试

来抵御网络中的各种攻击。

OSSIM 就是开源安全信息管理系统（Open Source Security Information Management），是 Alien Vault 公司开发的一个集中式检测网络威胁的网络安全平台。OSSIM 如名所述，是开源的，可以定制化。它可以升级成专业版 USM，可以提供更多的功能特性、存储空间、OTX 库和技术支持服务。它基于 Debian 操作系统，集成各种网络安全系统，例如 Snort、Suricata、HIDS 等，还可以集成其他的监控平台，如 Nagios 等，为网络监控提供一站式的统一服务。OSSIM 使用 B/S 架构，Web 服务器使用 Apache，数据库使用 MySQL，开发语言使用 PHP、Perl、C 等多种语言。

6.5.1 OSSIM 架构

OSSIM 系统主要分为如下三大块。

- 扫描：对系统漏洞进行扫描和分析，主要包含 Nmap、Openvas 等工具。
- 入侵检测（Intrusion Detection Scan，简称 IDS）：监视计算机系统和网络中发生的事件，对其进行安全分析，并生成入侵检测报告。在入侵检测中主要包含 Snort、OSSEC、Suricata、HIDS 等。
- 监控：是指监控 Linux、Windows、UNIX 等主机状态，以及交换机和路由器等网络设施状态，在系统运行异常的情况下，发送邮件和短信给网络安全人员。

如图 6-11 所示列出了 OSSIM 系统中的主要工具集。



图 6-11 OSSIM 工具集

6.5.2 安装与部署

2016年8月2日,开源安全软件提供商 AlienVault 发布了最新的 OSSIM 版本 AlienVault_OSSIM_64bits_5.3.0.iso。

OSSIM 镜像文件使用的是精简的 Debian 6 64 位操作系统。通过官方链接 (<https://www.alienvault.com/products/ossim/download>), 可以下载最新的 OSSIM 版本。

另外, 官网还提供了实例系统 (<https://demo.alienvault.com/ossim/>, 用户名 guest, 密码 alienvault)。

OSSIM 安装简便, 按照提示即可。安装时需要注意以下三点。

- 安装 OSSIM 内存建议 4GB 以上, 最好 8GB~16GB, CPU 分配双核以上, 硬盘分配 30GB 以上, 避免 OSSIM 启动不起来。如果使用商业版 USM, 建议参考官网数据。
- 在安装过程中, 需要设置语言、区域、键盘, 还有一个重要的“网络配置”。还需要静态设置 IP 地址、子网掩码、网关和域名服务器, 否则可能启动不了 OSSIM。
- 安装最后会要求设置 root 用户的密码, 等待安装结束即可。

6.5.3 OSSIM 控制台

OSSIM 分为前端的 Web 界面和后端的控制台, 安装好 OSSIM 后, 输入用户名和密码登录 OSSIM 后端的控制台, 可以对 OSSIM 系统进行设置, 主要包括防火墙、网络、插件、重启 OSSIM 服务器等。OSSIM 控制台如图 6-12 所示。

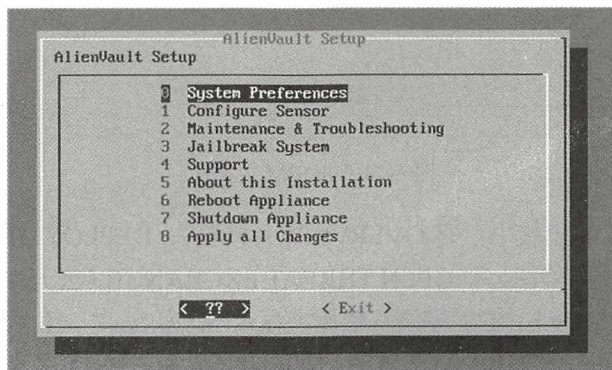


图 6-12 OSSIM 控制台

质量全面管控：从项目管理到容灾测试

6.5.4 Web 界面配置

OSSIM 提供 Web 界面来管理和查看网络情况。在浏览器中输入 “https://<IP 地址>”，这里的 IP 地址为安装过程中设置的 OSSIM 服务器静态 IP 地址。

在 OSSIM 的 Web 登录界面中（如果是第一次登录，则需要设置登录名、密码、邮箱等信息）输入用户名和密码，单击“LOGIN”按钮即可进入主界面，如图 6-13 所示。

进入 OSSIM 主界面，其显示如图 6-14 所示，主要包括 DASHBOARDS（仪表板）、ANALYSIS（分析）、ENVIRONMENT（环境）、REPORTS（报告）和 CONFIGURATION（配置）五部分。

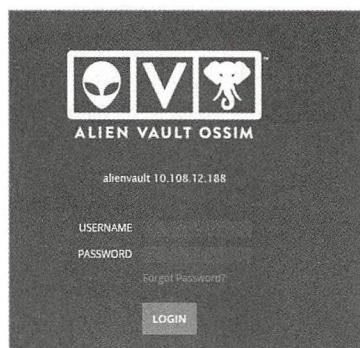


图 6-13 OSSIM 的 Web 登录界面

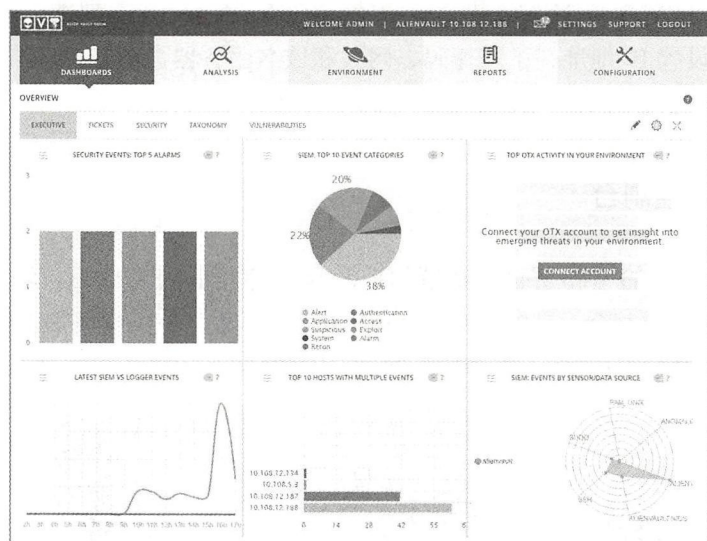


图 6-14 OSSIM 主界面

在 DASHBOARDS 模块中，有 OVERVIEW（概览）、DEPLOYMENT STATUS（部署状态）、RISK MAPS（风险图）和 OPEN THREAT EXCHANGE（公开威胁交换）四个菜单，如图 6-15 所示。DASHBOARDS 模块从整体上显示网络状况，一旦网络出现安全问题，能及时做出响应。其中，OPEN THREAT EXCHANGE 功能需要关联 OTX 账户之后才可以使用。

在 ANALYSIS 模块中，有 ALARMS（警报）、SECURITY EVENTS（SIEM）（安全事

件)、RAW LOGS (原始日志) 和 TICKETS (单据) 四个菜单, 如图 6-16 所示。ANALYSIS 模块主要用于分析网络状况, 识别网络风险和安全隐患。其中, 使用 TICKETS 菜单可以查看每台机器的状态。



图 6-15 DASHBOARDS 菜单



图 6-16 ANALYSIS 菜单

在 ENVIRONMENT 模块中, 有 ASSETS & GROUPS (资产和组)、VULNERABILITIES (漏洞)、NETFLOW (流量)、TRAFFIC CAPTURE (流量捕获)、AVAILABILITY (可用) 和 DETECTION (检测) 六个菜单, 如图 6-17 所示。

其中, ASSETS & GROUPS 菜单可以监控机器列表; VULNERABILITIES 菜单用于查看扫描漏洞; NETFLOW 菜单用于查看网络状况, 包含 TCP、UTD、ICMP 及其他网络资源; TRAFFIC CAPTURE 菜单用于抓取数据包; AVAILABILITY 菜单可以通过树型结构显示网络状况; DETECTION 菜单包含入侵检测, 并对入侵事件进行分类。

REPORTS 模块只有一个菜单 OVERVIEW, 包含报表的生成和导出, 也可以通过邮件发送 HTML 报告。

CONFIGURATION 模块主要提供配置系统、添加系统管理员、修改密码、设置语言和查看 OSSIM 服务器状态等功能。它有四个菜单: ADMINISTRATION (管理)、DEPLOYMENT (部署)、THREAT INTELLIGENCE (威胁智能) 和 OPEN THREAT EXCHANGE (公开威胁交换), 如图 6-18 所示。

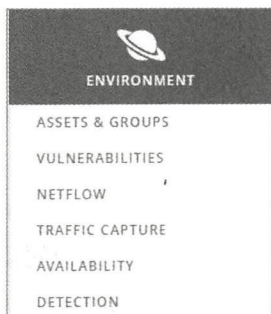


图 6-17 ENVIRONMENT 菜单



图 6-18 CONFIGURATION 菜单

质量全面管控：从项目管理到容灾测试

6.5.5 OSSIM 使用实战

下面简单介绍使用 OSSIM 进行主机漏洞扫描的步骤。

(1) 选择 ENVIRONMENT 模块的 VULNERABILITIES 菜单，在打开的漏洞扫描界面中切换到“SCAN JOBS”页面，如图 6-19 所示，然后单击“NEW SCAN JOB”按钮来新建扫描任务。

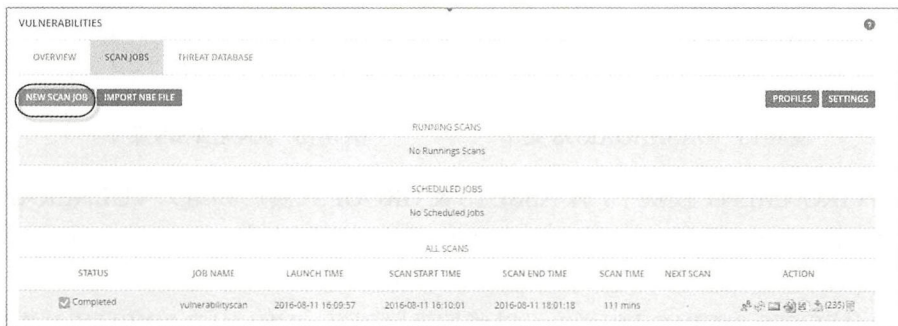


图 6-19 SCAN JOBS 页面

(2) 打开新建扫描页面，如图 6-20 所示，其中各选项含义如下。

- Job Name: 扫描任务名称。
- Select Sensor: 扫描的嗅探器。
- Profile: 扫描的类型，包括 Deep – Non destructive Full and Slow scan(深入)、Default – Non destructive Full and Fast scan(默认，扫描时系统可能会宕机崩溃)和 Ultimate – Full and Fast scan including Destructive tests(终极，包括压力测试)。
- Schedule Method: 扫描频率，可以选择 Immediately(马上)、Run Once(仅一次)、Daily(每天)、Day of the Week(每周)、Day of the Month(每月)和 Nth Week of the Month(每月第 N 周)。
- Asset List 列表框: 所有待扫描的主机，主机从右侧的资产树中可以选取。

设置完成后，单击“NEW JOB”按钮提交扫描任务。

(3) 创建扫描任务后，会根据扫描频率执行扫描任务。扫描执行时，可以查看任务下面每个主机的扫描进度和扫描出来的漏洞数量。如图 6-21 所示，页面中间罗列出的那些数字就是主机扫描出的漏洞数，分为 SERI(严重)、HIGH(高危)、MEDI(中等)、LOW(低危)和 INFO(信息)。其中，INFO 字段更多的是提示，表明执行的扫描次数。

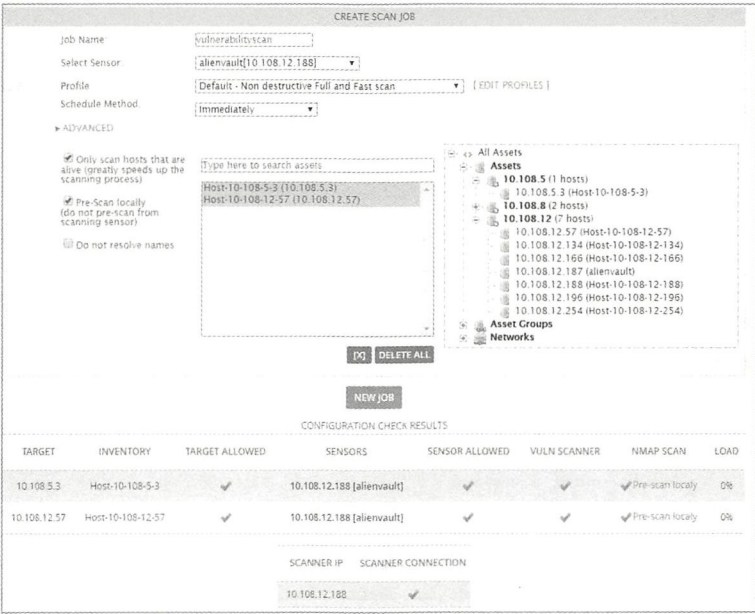



图 6-20 新建扫描页面

提示：在 ACTION 字段下，单击橙色方块可以停止当前的扫描任务，也可以取消循环的扫描任务。

NEW SCAN JOB

IMPORT NBE FILE

PROFILES

SETTINGS

1 RUNNING SCAN



JOB NAME	OWNER	SCAN TIME	CRASH	DOWN	UP	DOWN	UP	VULNS. TREND / SCAN PROGRESS	ACTION
 vulnerabilityscan	admin	RUN >55 mins	2	15	0	0	215	<div><div></div></div>	
	10.108.12.134 (Host-10-108-12-134)		0	4	0	0	27	<div><div></div></div> 100%	
	10.108.12.166 (Host-10-108-12-166)		1	3	0	0	20	<div><div></div></div> 100%	
	10.108.12.187 (alienvault)		0	1	0	0	43	<div><div></div></div> 100%	
	10.108.12.188 (Host-10-108-12-188)		0	2	0	0	44	<div><div></div></div> 100%	
	10.108.12.196 (Host-10-108-12-196)		0	2	0	0	17	<div><div></div></div> 100%	
	10.108.12.254 (Host-10-108-12-254)		0	0	0	0	5	<div><div></div></div> 100%	
	10.108.12.57 (Host-10-108-12-57)		1	3	0	0	59	<div><div></div></div> 94%	

图 6-21 OSSIM 扫描状态

(4) 当扫描结束后，也就是所有的主机显示扫描程度为 100%时，就可以在 VULNERABILITIES 的 OVERVIEW（概览）页面中查看到扫描结果，如图 6-22 所示。

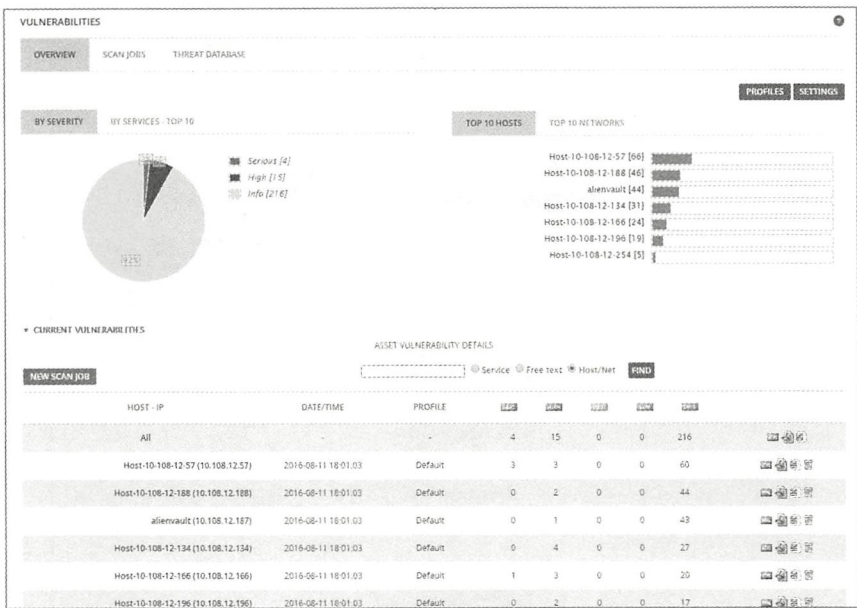


图 6-22 OSSIM 扫描结果

(5)扫描的结果还可以在 REPORTS 模块中查看。选择 REPORTS 模块中的 OVERVIEW 菜单，将显示各类报告，如表 6-9 所示。

表 6-9 OSSIM 报告及其内容

报告名	报告内容
Alarms	可根据时间和范围查看警报，例如攻击最多的前 10 台主机、被攻击最多的前 10 台主机、使用最多的前 10 个端口、最多的前 15 个警报和风险最多的前 10 个警报
Assets Details	资产信息，包括主机名、IP 和网络
Business & Compliance ISO PCI	需要遵守的业务规则，如 CIA、PCI-DSS、ISO27002 和 ISO27001 等
Geographic	资产地理分布图
Security Events	不同数据源的安全事件，例如防火墙、杀毒软件、数据库和应用服务器等
Tickets	关于警报、漏洞、事件等提出的请求事件
User Activity	Web 界面的用户操作，例如登录、生成报告等
Vulnerabilities Report	漏洞扫描报告，包括概况、各漏洞的具体信息

选择“Vulnerabilities Report”，单击“View Report”按钮可以显示漏洞报告，如图 6-23 所示。

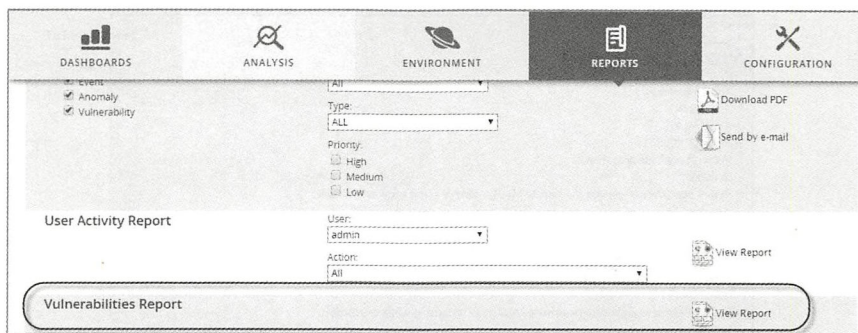


图 6-23 显示漏洞报告

打开漏洞报告,如图 6-24 所示,显示漏洞的概览和具体漏洞信息,本实例中的漏洞报告有 210 页。该报告显示各主机漏洞的情况、数量和严重性,数据和图 6-21 中的扫描结果是一致的。

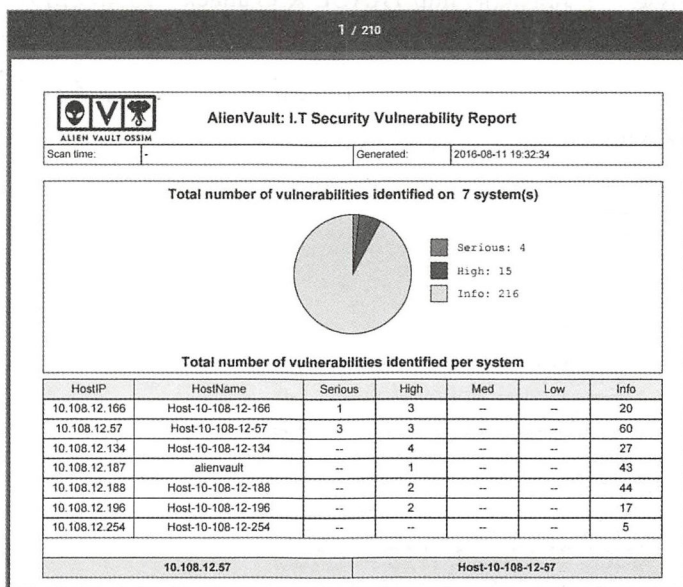


图 6-24 漏洞报告概览

如图 6-25 所示是其中的一个漏洞 MySQL weak password (MySQL 使用了弱密码),风险等级是 Serious (严重)。

该报告还列出了漏洞涉及的应用、端口、协议、脚本 ID 和解决方案等信息。

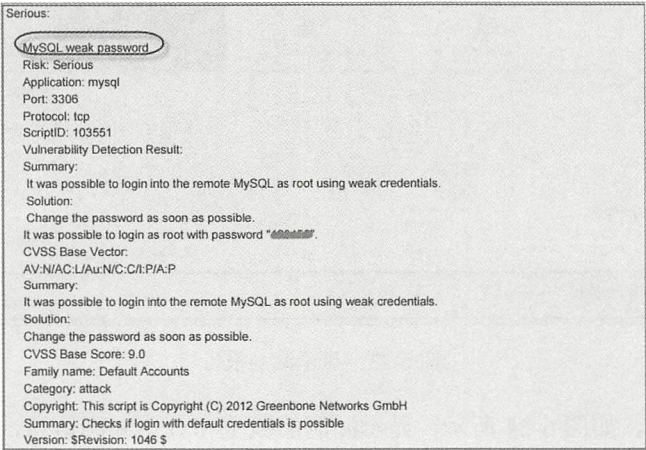


图 6-25 严重漏洞信息

如图 6-26 所示是一个高危漏洞 http TRACE XSS attack（跨站攻击），风险等级是 High（高危）。

这是来自于端口 8080 的应用，应该是 Tomcat 应用。应用服务器的调试函数 TRACE 和 TRACKBEIDAKAILE 被利用，造成 XSS 跨站攻击，所以需要禁用这些函数。

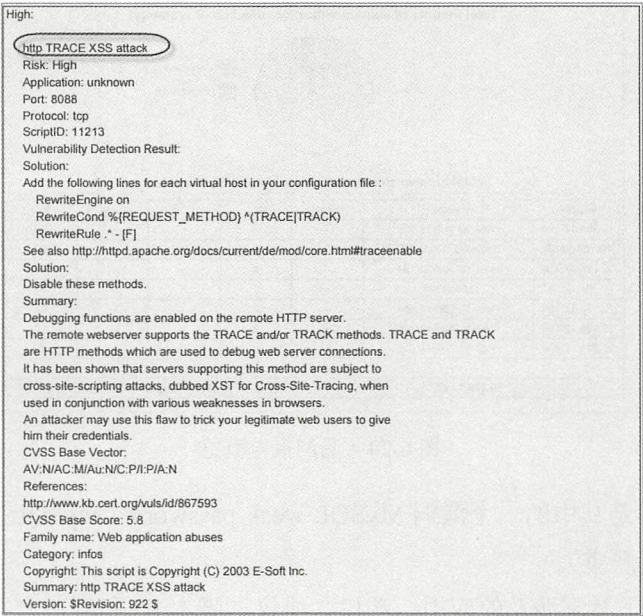


图 6-26 高危漏洞信息

除了漏洞,还会显示 Info 级别的信息,如图 6-27 所示。这条信息表明 OSSIM 中的 nmap 服务发现应用 ajp13 服务,其端口为 8009,但是却不能识别,以便后续分析。

```
Info:
Identify unknown services with nmap
Risk: Info
Application: ajp13
Port: 8009
Protocol: tcp
ScriptID: 66286
Vulnerability Detection Result:
Nmap service detection result for this port: ajp13
Summary:
This plugin performs service detection by launching nmap's
service probe against ports running unidentified services.
Description :
This plugin is a complement of find_service.nasl. It launches
nmap -sV (probe requests) against ports that are running
unidentified services.
CVSS Base Vector:
AV:N/AC:L/Au:N/C:N/I:N/A:N
CVSS Base Score: 0.0
Family name: Service detection
Category: infos
Copyright: Copyright (c) 2009 E-Soft Inc. http://www.securityspace.com
Summary: Launches nmap -sV against ports running unidentified services
Version: $Revision: 329 $
```

图 6-27 Info 信息

6.6 要点回顾

信息安全无处不在。通过信息收集和信息安全分析,制定安全测试的策略,对口令、Web 服务器、密码安全和会话进行安全分析,找到安全漏洞,分析后提出解决方案来加固系统,使系统在受到外界攻击后有相应的防御措施。

第 7 章

自动化测试基础

在传统的功能测试中，测试人员首先根据需求功能点编写测试用例，再对测试用例进行评审。评审通过后，根据测试用例中描述的步骤来执行测试，把实际测试结果和预期结果进行对比，从而验证被测系统是否满足功能需求。在这个过程中如果采用手工方式执行测试用例，速度必然很慢，同时，测试人员如果需要对同一个问题进行多次验证，由于经常做一些重复性劳动，会很快产生厌倦，造成测试效率低下。在测试用例非常多的情况下，执行一轮完整的测试用例花费的时间是相当高的。

为了解决功能测试执行效率低下的情况，提出了自动化测试，使自动化测试部分代替功能测试，那么测试用例的执行人从人力变成机器，机器可以 24 小时不间断地执行，提高了测试执行的效率，通过多台机器并行执行，大大缩短了执行的时间，提高单位时间内测试执行的覆盖率。

俗话说“工欲善其事，必先利其器”，在开始自动化测试前需要了解常用的自动化测试工具。自动化测试工具经过这几年的发展越来越多，有商业版、开源版、自主研发版等，根据不同类型的系统可以使用不同的自动化工具，例如：Web 界面可以使用 Selenium、QTP、Cucumber、RTF 等；移动端测试可以使用 Appium、Monkey、EarlGrey、Calabash 等；接口测试应该根据接口不同、协议不同使用不同的测试工具，如 JMeter、SoapUI、HttpClient 等。如果以上工具都不能满足读者需求，可以通过编程语言自主开发自动化工具来完成自动化测试任务，或者引用一些开源的自动化框架如百度的 Cafe、淘宝的 Macaca 和 AutoMan 以及 Nokia Siemens Networks 公司开发的 Robot Framework 等。

学习自动化测试之前需要熟悉至少一门开发语言，如 Java 语言，因为仅仅借助自动化

工具自动录制的脚本，不能满足实际需求，其根本原因是业务系统为了提升用户体验，造成界面变化频繁，因此需要编写自动化脚本来满足界面的不断变化。自动化测试涉及的知识面比较广，本章将从如下几个方面介绍自动化测试：

- 自动化测试实施流程
- 自动化测试工具
- 标记语言 HTML、XML、XHTML 等
- Web 自动化测试
- Mock 测试
- HTTP 协议测试
- TestNG 测试框架

7.1 自动化基础

在软件测试过程中，为了节省人力资源、时间和费用，提高测试效率，引入了自动化测试。自动化测试是软件测试的一个重要组成部分。自动化测试是把手工执行测试的过程转换为机器执行测试的过程，用机器语言来模拟自然人的操作。自动化测试人员根据评审通过后的测试用例来编写和执行自动化脚本，最终产出测试报告，分析测试报告并根据测试报告评估被测系统的质量。一般来说，自动化测试主要应用在以下场景。

- **冒烟测试：**冒烟测试（SmokeTest）是在新版本的软件上执行关键性的测试用例，保证新的程序代码不出故障。通过冒烟测试之后，可以继续执行更多其他种类的测试。
- **回归测试：**回归测试（Regression Test）主要是验证新发布的功能或修复的 Bug 有没有对现有的系统功能造成影响。
- **兼容性测试：**兼容性测试（Compatibility Test）主要是验证被测试系统在不同的硬件平台、不同的操作系统、不同的应用软件之间和不同的网络等环境中是否能够正常地运行。常规的兼容性测试可以分为浏览器兼容性测试、操作系统兼容性测试和移动设备的兼容性。

7.1.1 自动化测试流程

自动化测试过程与软件开发过程类似，一般包含以下几个步骤：前期通过需求调研和可行性分析，确定是否展开自动化测试，如果确定开展自动化测试，需要进行测试需求分析并制定相应测试计划；中期设计自动化测试用例，完成自动化脚本的开发联调，运行测试脚本提交阶段性测试报告并对测试报告进行分析；后期需要对测试脚本和测试框架不断优化，从而满足不同的业务需求是一个持续维护的阶段。如图 7-1 所示描述的是一般自动化测试过程的基本流程。

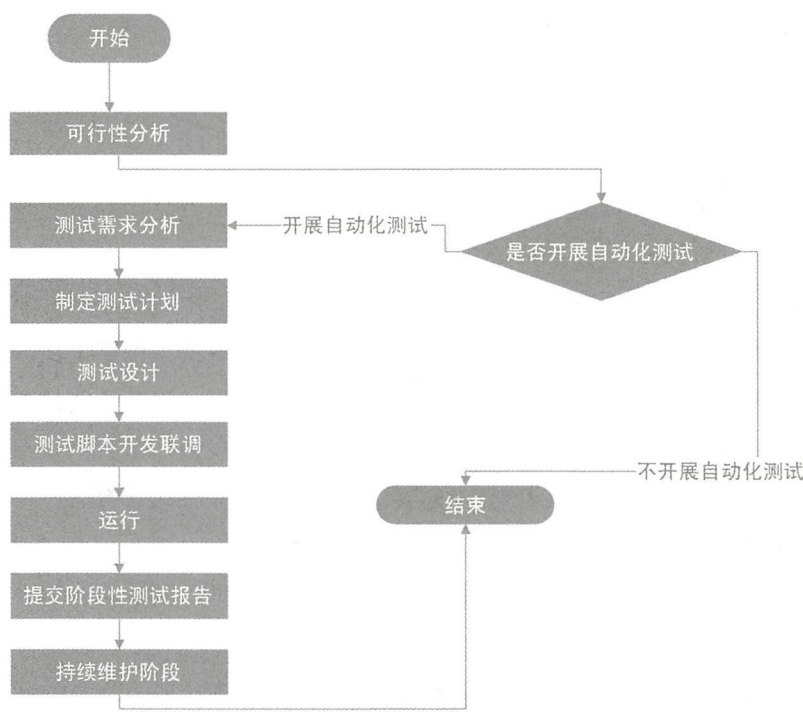


图 7-1 自动化测试基本流程

1. 可行性分析阶段

可行性分析阶段主要确定是否开展自动化测试，可以从如下几个方面进行考虑。

- 项目周期时间长：自动化测试项目从立项开始，自动化测试框架的设计和测试脚本的编写与调试都需要经过相当长的周期才能完成。项目前期花费的人力和财力

都比较大。如果项目周期过短，就没有足够的时间来支撑这一过程，开发好的脚本也不能最大限度地复用，这就失去了自动化测试的价值。一般来说，可以考虑项目周期在 3 个月以上；

- 页面对象容易识别：如果被测系统页面对象不能识别，将不利于后期自动化脚本的开发。前期需要对被测系统有所了解，决定所选择的测试工具是否适用被测系统；
- 软件需求变更不频繁：测试脚本的稳定性决定了自动化测试的维护成本。如果软件需求变动过于频繁，测试人员需要根据变动的需求来更新测试用例以及相关的测试脚本，而脚本的维护本身就是一个代码开发的过程，需要进行修改和调试。必要的时候还要修改自动化测试的框架。如果自动化测试所花费的成本高于利用自动化测试节省的测试成本，那便失去了自动化测试的意义。

2. 测试需求分析阶段

在这个阶段主要熟悉被测系统的业务流程，分析测试需求，为后期的测试计划做准备。

- 获取系统开发技术文档，熟悉系统架构及开发使用的技术。比如系统是 B/S 还是 C/S，系统后端是 Java 还是 .Net 开发的；
- 根据已有的手工测试用例来了解系统业务的流程。

3. 制定测试计划阶段

确定开展自动化测试以后，需要在自动化测试实施过程中进行规划，保证自动化测试任务的顺利完成，此阶段主要工作如下所示。

- 资源：包含自动化实施人员、业务支持人员和业务系统相关文档；
- 测试范围：自动化测试用例需要覆盖的范围（如冒烟测试用例，回归用例等）；
- 测试工具的选择：根据被测试系统的特点，选择测试工具；
- 测试进度：各个时间段需要完成的任务状态；
- 交付物：每个阶段完成后，有哪些交付物。

4. 测试设计阶段

此阶段主要由自动化框架开发人员和自动化脚本开发人员共同完成。

- 自动化框架开发人员需要设计自动化测试框架。在设计自动化测试框架时，需要考虑到脚本的复用性，以及秉承数据、业务和脚本相分离的宗旨，方便自动化脚本开发人员进行自动化测试脚本开发；

- 自动化脚本开发人员需要把手工用例转换为自动化测试用例，有时也需要根据需求补充自动化用例，完成系统自动化用例的设计开发。

5. 脚本开发联调阶段

根据自动化用例开发测试脚本，尽量保持测试脚本的特性满足如下所示。

- 可复用性：把一个功能写成一个模块，以便再次需要相同功能的时候，可以直接使用，而不用重新开发。
- 健壮性：处理异常的能力，在出现异常情况下能够独立处理异常，保证脚本顺利运行。
- 稳定性：脚本在运行过程中，尽可能不出现异常、错误、中断等问题。
- 模块化：把一个复杂的模块分解成相互独立的小模块，这些小的模块是可以自由组合更换。
- 跨平台性：测试脚本应该兼任不同的平台和浏览器，尽量保证测试脚本不受平台的变更影响。

6. 运行阶段

搭建好被测系统环境，部署了自动化测试脚本，就可以开始对系统进行自动化测试，并自动记录测试结果，然后根据测试结果进行分析。在本阶段应该考虑以下 4 种情况。

- 被测系统：如果是被测系统的缺陷造成系统不能使用，应提交相应的缺陷报告。
- 测试脚本：因为脚本是由测试人员编写的，难免出现相应的脚本问题，应该修改失败的脚本。
- 测试数据：如果是相应测试数据导致自动化测试失败，应修改相应数据。
- 测试环境：在执行自动化脚本时，出现环境问题，可以采用场景恢复、重新执行失败的测试用例等策略解决此类问题。

7. 提交阶段性测试结果

根据运行结果，编写测试报告，总结本次运行结果，衡量被测系统是否通过自动化测试。测试报告主要包含运行环境、覆盖用例数、失败用例数、成功用例数、执行时间和执行人员等指标。

8. 持续维护阶段

自动化测试需要经历一个长期的过程。每次版本的变更后，都需要不断地维护相应的

脚本，对测试脚本进行优化和持续更新，更好地帮助手工测试人员提高工作效率适应变更后的系统，保证系统平稳上线。

7.1.2 自动化测试特点

简单地讲，自动化测试就是把功能测试用例进行脚本化，然后通过执行机器批量运行脚本，最终产生一份自动化测试报告和对自动化测试报告进行分析的过程。自动化测试主要包含以下7大特点，如图7-2所示。

- 覆盖率高：覆盖率一般指覆盖项目功能点或者手工用例的比例。覆盖率越高，说明自动化测试覆盖面越广，能够保证系统上线后稳定运行，降低潜在的风险。
- 节省人力成本：由于项目的变更迭代，引起了大量的回归测试，自动化测试能有效地减少回归测试期间的人力投入用更短的时间完成回归测试。
- 保证基本功能的正确性：对于系统的冒烟测试，通过自动化测试方式，能对系统的基本功能更加放心。
- 测试脚本和测试数据复用性：测试脚本和测试数据可以多次重复执行。
- 测试时间短：由于自动化测试比人工测试快，所以可以在有限的测试时间实现快速的回归测试。
- 测试并行：为了提高测试执行时间，可以使用多台执行机并行测试，缩短测试时间。
- 无人值守：自动化测试可以长期不间断地运行。

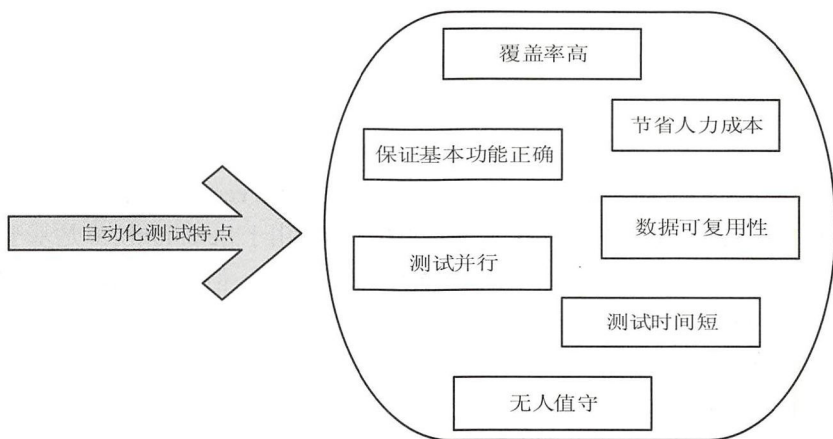


图 7-2 自动化测试特点

7.1.3 自动化测试工具

软件的质量越来越被重视，因此软件测试工作也随之被重视。自动化测试属于软件测试的一种类型，其开展需要测试工具的支撑。自动化测试工具有开源的，也有商业版的。商业版的自动化测试工具需要购买软件许可证，其价格一般比较昂贵，在选择自动化测试工具时，应该把此因素考虑进去。目前市场上主流的自动化测试工具有 Selenium、QTP、Cucumber、EarlGrey 等。下面来简要地介绍以下几款主流的自动化测试工具。

1. Selenium 1.0

Selenium 1.0 是 ThoughtWorks 公司开发的一套基于 Web 应用的测试工具，直接运行在浏览器中，模拟用户的操作，被用于单元测试、冒烟测试、集成测试和验收测试中，它可以运行在各种主流的浏览器和操作系统中。Selenium 1.0 主要由 3 部分组成，如图 7-3 所示。

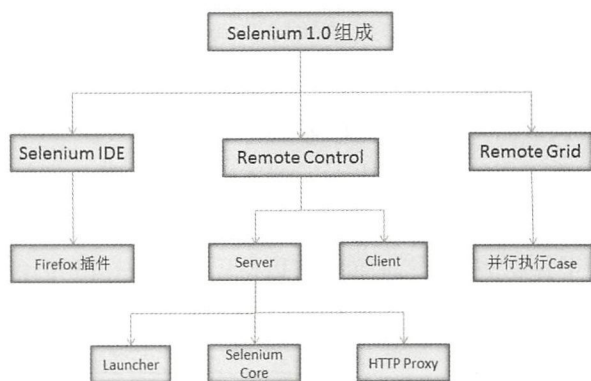


图 7-3 Selenium1.0 组件

Selenium 1.0 主要包含 Selenium Core、Launcher 和 Http Proxy，使用 Selenium1.0 首先需要启动 Selenium Server 端。Selenium Core 被 Selenium Server 嵌入到浏览器中，通过调用 Selenium Core 的 JavaScript 函数来实现对浏览器的操作。由于 Selenium 1.0 是基于 JavaScript 实现的，因此可以很好地运行在支持 JavaScript 的浏览器上。但是 JavaScript 有严格的安全限制，从而导致 Selenium 1.0 不能处理如下事件：

- 执行本机键盘和鼠标事件；
- 同源策略 XSS/HTTP (S)；
- 弹出框和对话框；
- 基于身份认证，自签名的证书和文件上传、下载。

2. Selenium 2.0

Selenium 2.0 合并了 Selenium 1.0 和 WebDriver。WebDriver 是一款开源 Web 自动化测试工具，该工具发行于 2011 年由 Simon Stewart 创建。Selenium 2.0 吸取了 Selenium 1.0 和 WebDriver 的优点，主要体现在以下几个方面。

- Selenium 2.0 对不同浏览器的处理方式不同。Selenium 2.0 不使用 JavaScript 沙盒，而是选择浏览器最容易接受的语言来处理，回避了 JavaScript 的安全限制，克服了 Selenium 1.0 的先天不足。
- Selenium 2.0 继承了 WebDriver 清晰的面向对象 API，并能以最佳的方式与浏览器进行交互，方便测试人员开发脚本。
- Selenium 2.0 可以直接操作 HTML Unit 驱动，通过 HTML Unit 在系统内存中迅速执行，提高执行效率，为弹出窗口提供更好的支持。
- 由于 Selenium 2.0 能够调用操作系统 API，所以能有效控制本机的键盘与鼠标事件。
- 运行 Selenium 2.0 不再需要启动 Server 端。
- 支持 Android（Selendroid）和 iPhone（Appium）的移动应用测试。

3. QTP（Quicktest Professional）

QTP 是惠普公司提供的一款商业版自动化测试工具，主要应用于软件测试中的功能测试和回归测试，可以运行在 B/S、C/S 等终端的测试系统。用户可以使用其自带的录制功能来录制脚本，也可以使用其回放功能来回放录制脚本。该工具主要特点如下所示：

- 该工具侧重功能的回归测试，提供了诸多插件支持 Web、VB.Net、Java、SAP 等应用，分别用于各种产品的回归测试。
- QTP 支持的脚本是 VBS，对测试人员来说学习比较简单。
- QTP 支持录制回放功能，录制的脚本可以拿来作为编写测试脚本的基础。
- QTP 提供 Object Spy，可以用来抓取测试对象的属性。
- QTP 提供 Excel 形式的数据表格 DataTable，可以用来存放测试数据和参数。
- 一个测试流程中使用多个 QTP 的 Action 来组织测试流程。
- QTP 为每个测试提供一个测试结果，包括 Passed、Failed、Done 和 Warning 几种状态类型。

4. Cucumber

Cucumber 是目前非常流行的面向行为的自动化测试框架，它具有跨平台性，可以使用



Java、C#和 Python 等语言编写测试脚本，大部分 IDE 都有对应的集成插件，便于安装使用。Cucumber 使用一个特性文档来描述用例，特性文档采用近似于普通文本的自然语言写成，可以支持中文和英文两种语言，使用自然语言编写的用例通俗易懂和易于编写，即使是非技术人员也可以编写测试用例和读懂测试用例。

5. EarlGrey

EarlGrey 是 Google 刚刚发布的一款测试 iOS 应用的开源自动化工具，在这之前可以使用 Appium 等测试工具实现移动端的自动化测试。EarlGrey 的代码已经托管于 GitHub，该工具使用 Objective-C 编写而成，可很好地支持和模拟 iOS 上的设备和应用。

7.1.4 标记语言介绍

标记语言是一种将文本及文本相关的信息结合起来，展现出关于文档结构和数据处理细节的编码，被广泛用于网页和网络应用程序，是互联网不可或缺的一部分。标记语言主要包含以下几种类型。

- HTML：全称 Hyper-Text Markup Language，超文本标记语言。
- XML：全称 Extensible Markup Language，可扩展标记语言。
- XPATH：描述 XML 路径的表达式。
- XHTML：全称 Extensible HyperText Markup Language，可延伸超文本标记语言。

1. HTML

HTML 是超文本标记语言，是由一套标记符号和语法规则组成，用于表现数据。HTML 用一对带尖括号“<>”表示超文本标记。这些超文本标记一般是成对出现的，用带斜杠（/）的元素表示结束。HTML 语言将所有要显示出来的文字、图形、声音等要素按照一定的标准排放，形成一定的标题、段落、列表等单元。超文本传输协议（HTTP）规定了浏览器在运行 HTML 文档时所遵循的规则和进行的操作，所以使用 HTML 语言描述的文件可以通过浏览器显示出效果。一个基本的 HTML 文档代码如下：

```
<HTML>
<HEAD>
<TITLE>一个简单的 HTML</TITLE>
</HEAD>
<BODY>
<CENTER>
```





```
<H4>我的主页</H4>
<HR>
<FONT SIZE=2>
    这是我第一次做主页
</FONT>
</CENTER>
</BODY>
</HTML>
```

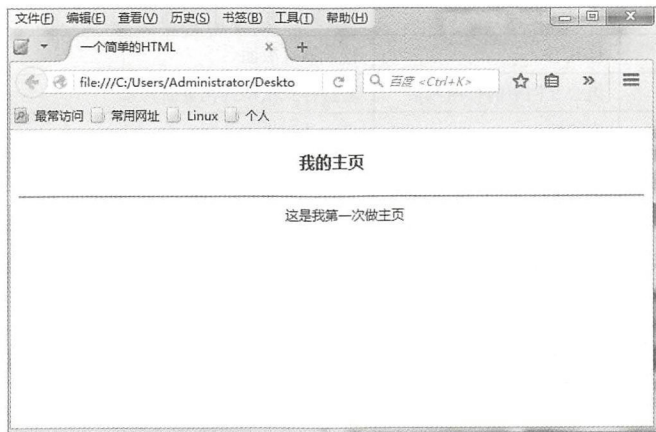


图 7-4 HTML 实例

将该文档存为 index.html，使用 Web 浏览器打开，可以查看其运行效果，如图 7-4 所示。

自动化测试过程，经常抓取网页对象，并且对该网页对象进行单击、双击和光标定位等操作，因此熟悉 HTML 是学习自动化测试的基础。常用的 HTML 标记如表 7-1 所示。

表 7-1 常用的 HTML 标记

标记	意义	作用
<HTML>	文件声明	让浏览器知道这是 HTML 文件
<HEAD>	开头	提供文件整体信息
<P>	段落标记	浏览器会自动地在段落的前后添加空行
 	换行标记	换行
<HR>	水平线	插入一条水平线
<TITLE>	标题	定义文件标题，将显示于浏览顶端
<BODY>	定义文档的主体	定义网页的主体部分，是用户可以看到的内容，如文字、图片、视频等
<CENTER>	居中	文字、图片、表格等显示于中间





续表

标记	意义	作用
<DIV>	设定文档内大块元素的背景和结构	可以结合 Class 样式表对大块元素进行格式化
	字形标记	设定字形、大小、颜色
	顺序清单	将数字、字母顺序排列
	清单项目	每一标记标示一项清单项目
<A>	定义超链接	用于从一张页面链接到另外一张页面
<STYLE>	样式表	控制网页版面
	组合文档中的行内元素	通过样式来格式化页面元素
<TABLE>	表格标记	设定该表格的各项参数
<CAPTION>	表格标题	为表格定义一个标题，通常这个标题会被居中于表格之上
<TR>	表格列	设定该表格的列
<TD>	表格栏	设定该表格的栏
<TH>	单元格	定义表格内的表头单元格，可以用 colspan、rowspan 属性控制单元格
<FORM>	表单标记	决定单一表单的运作模式
<TEXTAREA>	文字区块	输入较大量文字
<INPUT>	输入标记	决定输入形式
<SELECT>	选择标记	建立 pop-up 卷动清单
<OPTION>	选项	每一标记标示一个选项
<FRAMESET>	框架集	设定框架集
<FRAME>	设定框架	定义 frameset 中的一个特定的窗口（框架）
<IFRAME>	页内框架	于网页中间嵌入框架
	图形标记	用以插入图形及设定图形属性
<LINK>	关系定义	定义该文件与其他 URL 的关系
<META>	提供有关页面的元信息	主要针对搜索引擎和更新频度的描述
<!--注释-->	注释标记	为文件加上说明，但不被显示

2. XML

XML 指可扩展标记语言（Extensible Markup Language）。XML 被用来作为多种数据源间进行数据的交换或存储，广泛应用于互联网，同时也很容易学习。

XML 文档定义方式有：文档类型定义（DTD）和 XML Schema。DTD 定义了文档的整体结构以及文档的语法；XML Schema 定义可以出现在 XML 文档里的元素和元素的个数、类型。XML 文档类似一种树结构，它从根节点开始，向四周扩散，形成枝叶。如下代





码展示了一个标准的 XML 文档结构实例。

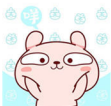
```
<Store>
  <book name="测试的艺术">
    <title country="CH">Test</title>
    <author>张三</author>
    <year>2015</year>
    <price>30.00</price>
  </book>
  <book name="计算机应用">
    <title country="CH">Computer</title>
    <author>胡浩</author>
    <year>2017</year>
    <price>39.99</price>
  </book>
  <book name="JAVA 编程思想">
    <title country="CH">JAVA</title>
    <author>Erik</author>
    <year>2013</year>
    <price>99.99</price>
  </book>
</Store>
```

实例中根元素是<Store>，所有的<book>元素被包含在<Store>中，<book>元素的 4 个子元素分别是<title>、<author>、<year>、<price>。这里注意一点，XML 元素是大小写敏感的。通过 XML 文档能看出数据的基本结构，简化了网络中数据交换和表示，使得数据、代码和表示相分离，并作为网络数据交换的标准格式，因此它常被称为智能数据文档。

3. XHTML

XHTML 是以 XML 应用的方式定义的 HTML，是更纯净、有规则的 HTML。相比于 HTML 中<!DOCTYPE...>，XHTML 是强制性的，<html>、<head>、<title>以及<body>也是强制性的。HTML 和 XHTML 还有一些元素和语法上有所不同，有兴趣的读者可以查阅相关资料，本书不再详细介绍。将前面 HTML 的例子修改成一个标准的 XHTML 文档结构代码如下：

```
!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```





```
<head>
<title>一个简单的 HTML</title>
</head>
<body>
  <center>
    <h4>我的主页</h4>
    <hr>
    <font size="2">
      这是我第一次做主页 XHTML
    </font>
    </hr>
  </center>
</body>
</html>
```

从此例子可以看出，XHTML 对于元素要求参照 XML，必须拥有根元素；XHTML 文档必须有始有尾，不可以只有标签头<>，无标签尾</>。另外，元素和属性要求都是小写的，属性必须要加双引号。

将文件保存成 first.xhtml，在浏览器中运行 XHTML 代码，显示如图 7-5 所示，跟原本的 HTML 还是比较一致的。



图 7-5 XHTML 示例

4. XPath

XPath 即为 XML 路径语言，用来查找 XML 文档中信息。XPath 是基于 XML 的树状结构，XPath 包含了一个标准的函数库，通过 XML 文档中的元素和属性找寻节点的能力。因此也可以说它是一种信息查找语言。

XPath 使用路径表达式来选取 XML 文档中的节点或节点集，节点是沿着路径来选取的。表 7-2 列举出了常用的 XPath 常用路径表达式。

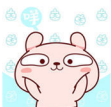




表 7-2 XPath 常用路径表达式

表达式	描述
nodeName	选取此节点的所有子节点
/	从根节点选取
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置
.	选取当前节点
..	选取当前节点的父节点
@	选取属性

根据 7.2 小节的 XML 文档结构实例，表 7-3 列出了关于 XPath 表达式路径和对应路径表达式搜索出来的结果。

表 7-3 XPath 表达式实例

路径表达式	结果
Store	选取 Store 元素的所有子节点
/Store	选取根元素 Store
Store/book	选取属于 Store 的子元素的所有 book 元素
//book	选取所有 book 子元素，而不管它们在文档中的位置
Store//book	选择属于 Store 元素的后代的所有 book 元素
//@country	选取名为 lang 的所有属性
/Store/book[1]	选取属于 Store 子元素的第一个 book 元素
/Store/book[last()]	选取属于 Store 子元素的最后一个 book 元素
//title[@country='CH']	选取所有 title 元素，且该元素含有 country 属性并且其属性值为 CH
//title[@country]	选取所有拥有名为 country 的属性的 title 元素
/Store/*	选取 Store 元素的所有子元素
//*	选取文档中的所有元素
//title[@*]	选取所有带有属性的 title 元素

7.2 Web 自动化测试

常见的 Web 自动化测试，主要使用 IE、Chrome 和 Firefox 等浏览器运行被测系统，类似于手工测试，首先需要找到单击的按钮位置，在自动化测试中称为元素定位，主要使用浏览器插件对元素定位，找到相应的元素。找到元素可以使用 Selenium IDE 录制脚本，或者使用 Selenium 的 API 编写测试脚本，然后对元素进行单击、下拉、输入等动作后，检查





预期结果和实践结果是否一致，代替繁琐的手工测试。

7.2.1 元素定位

Web 自动化测试脚本开发中，只有定位到页面元素，才能对页面元素进行操作。因此，自动化脚本开发人员需要捕获页面元素，并使用代码描述元素位置。由于 Selenium 自动化测试工具未提供类似 QTP 可以抓取页面元素的工具，所以页面元素的捕获需要依赖于各类 Web 浏览器提供的插件，通过这些插件抓取页面元素的源码，再分析这些源码进行元素定位。本小节主要介绍一些页面元素定位的工具，通过这些工具实现页面元素的定位，并结合 WebDriver 的 API 来实现页面元素的定位。

常用的 Web 浏览器一般为 IE、Chrome、Firefox、Safari，B/S 架构的 PC 客户端通常以这些 Web 浏览器为载体，其中移动端 HTML 5 项目可以使用 Chrome 的 toggle device mode 模拟。页面元素的定位推荐使用 Firefox 浏览器的 Firebug 和 FirePath 插件，或者 Chrome 浏览器中提供的开发者工具来定位元素。若被测系统不支持这两种浏览器，则可使用 IE 的开发人员工具。Web 页面元素定位不会因浏览器的不同而有所不同，因此可以使用不同的浏览器来进行元素定位。

1. Firefox 插件

(1) 下载并安装 Firefox 浏览器，然后安装 Firebug 和 FirePath 这两个插件，方便识别 XPath。

(2) 使用 Firefox 浏览器，进入被测系统页面，按 F12 键调出 Firebug 插件，Firebug 插件界面显示如图 7-6 所示。

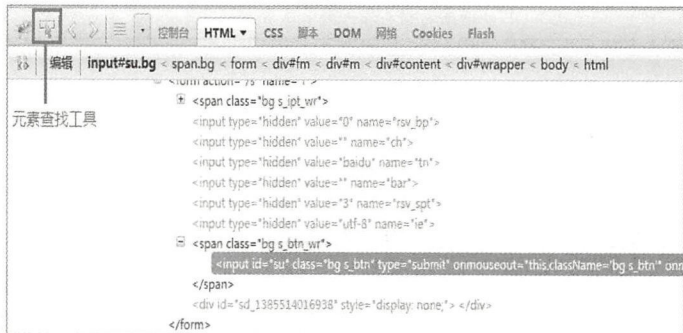


图 7-6 Firebug 插件






- (3) 单击元素查找工具图标，选中一个页面元素，即可查看定位到该元素的页面代码。
- (4) 将鼠标指针悬停在 Firebug 中的某一行代码上，相应的页面元素会高亮显示，如图 7-7 所示。



图 7-7 Firebug 元素高亮

- (5) FirePath 是 Firefox 的一个插件，可以通过该插件查看 XPath 的值。通过 Firefox 浏览器选择菜单项“工具→附加组件”，搜索“FirePath”后，下载并安装该插件，如图 7-8 所示。



图 7-8 FirePath 插件





(6) 重启 Firefox 浏览器，在网页上选择要查看元素的 XPath，右击，在弹出的快捷菜单中单击“Inspect in FirePath”选项，将出现该元素 XPath 的值，如图 7-9 所示。



图 7-9 单击“Inspect in FirePath”选项

(7) 通过 FirePath，可以查看到 XPath 结果是“//*[@id='su']”，如图 7-10 所示。




图 7-10 使用 FirePath 查看的 XPath 值

2. Chrome 插件

(1) 打开 Chrome 浏览器，进入被测页面，按 F12 键调出开发者工具，如图 7-11 所示。



(2) 单击上图中的放大镜工具，选择一个页面元素，即可查看定位到该元素的页面代码。

(3) 当鼠标指针悬停在某一行代码上，相应的页面元素会高亮显示。

(4) 如果要查看该元素的 XPath，选择某一行代码，然后右击，在弹出的快捷菜单中选择“Copy XPath”选项即可复制该元素的 XPath，如图 7-12 所示。

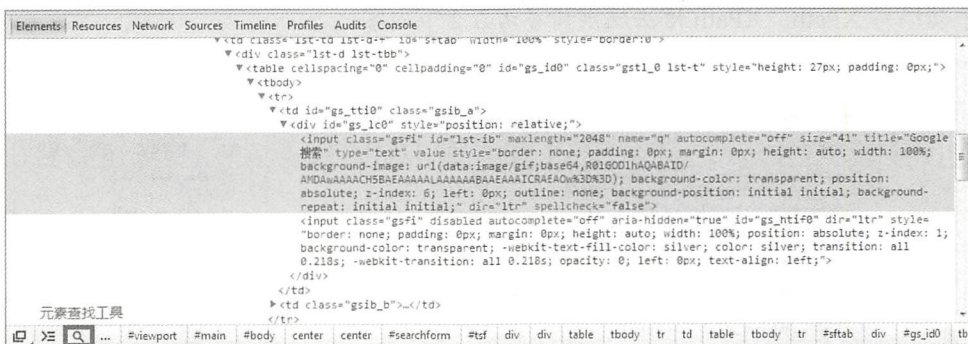


图 7-11 Chrome 开发者工具

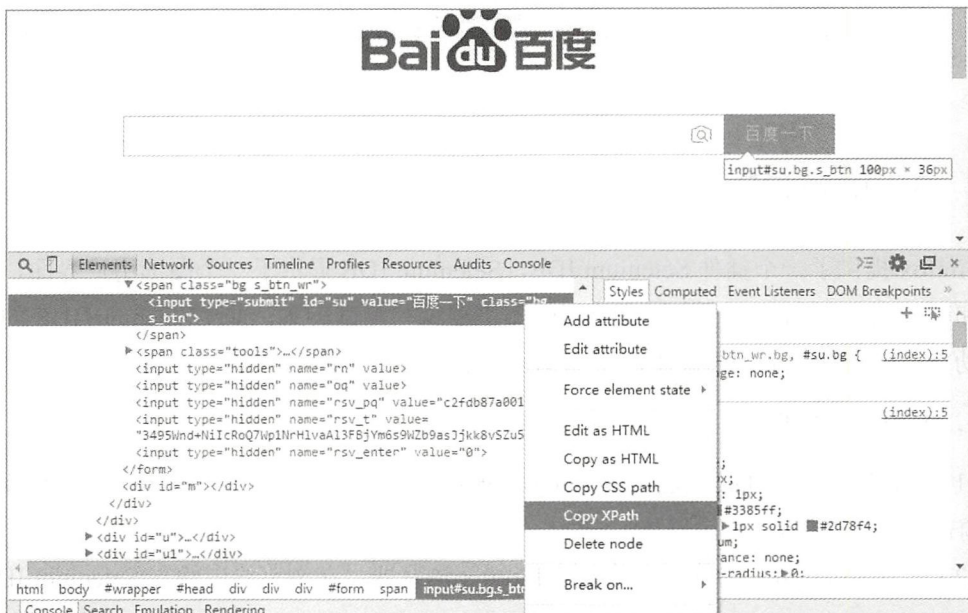


图 7-12 复制 XPath

3. Internet Explorer 插件

(1) 打开 IE，进入被测页面，按 F12 键调出开发人员工具，如图 7-13 所示。IE8 以上版本的浏览器中已经内置了开发人员工具。

(2) 单击小箭头图标，选择一个页面元素，即可查看定位到该元素的页面代码。

提示：IE 开发人员工具并没有向 Chrome 那样查看 XPath 的插件，需要用户根据网页代码手动编写 XPath。XPath 语法规则请参考 7.2 节内容。

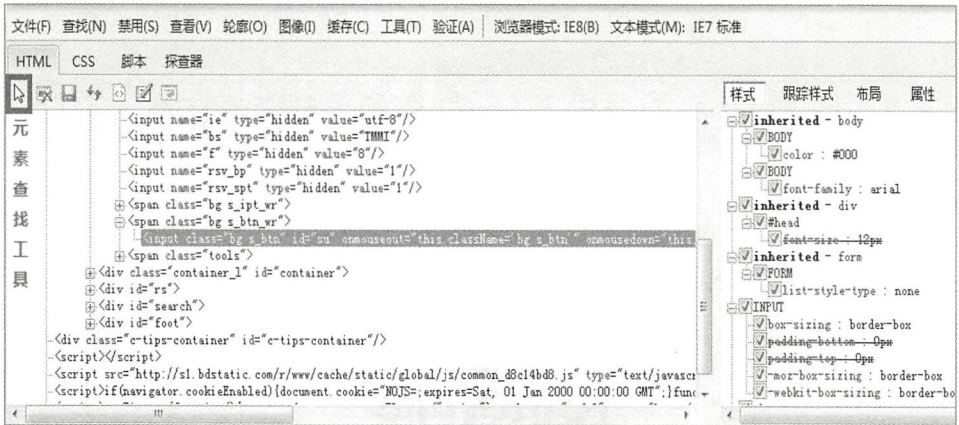


图 7-13 IE 开发人员工具

7.2.2 Selenium IDE

Firefox 提供了一个插件 Selenium IDE，Selenium IDE 提供了一个图形用户界面，能够录制和回放用户在 Firefox 中用户的行为。不需要编写测试脚本，通过 Selenium IDE 的导出脚本功能，能自动生成测试脚本。下面描述 Selenium IDE 的操作主要过程。

1. 使用 IDE 插件

(1) 由于 Selenium IDE 只能用于 Firefox 浏览器中，不支持其他浏览器，在安装 Selenium IDE 前需要先安装 Firefox 浏览器。

(2) 安装完 Firefox 浏览器后并打开 Firefox 浏览器，在浏览器中输入 Selenium IDE 插件地址：<https://addons.mozilla.org/en-US/firefox/addon/selenium-ide-button/>。界面如图 7-14 所示，单击“Add to Firefox”按钮添加插件到 Firefox 浏览器。

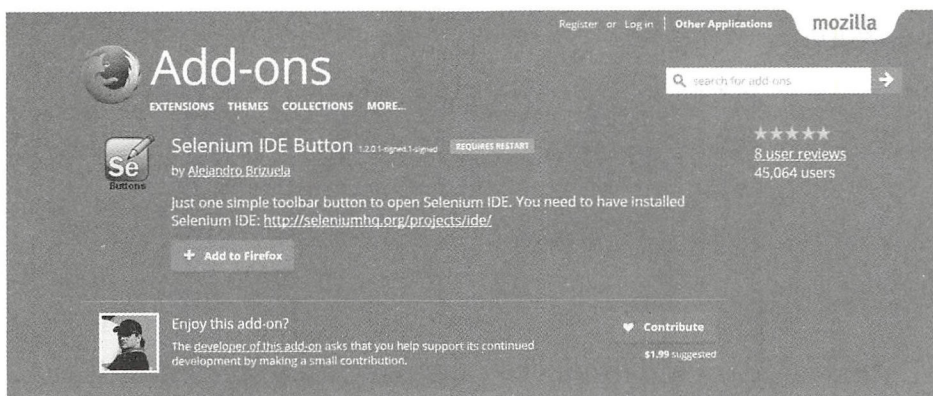


图 7-14 Selenium IDE 插件

(3) 安装完成后重启浏览器，打开浏览器，选择菜单项“查看”，单击“工具栏”下面的“定制”选项，可以查看到 Selenium IDE Button 插件，如图 7-15 所示，表示该插件安装成功。也可以通过鼠标选中 Selenium IDE Button 插件后，拖拽到常用菜单栏中，方便以后使用。

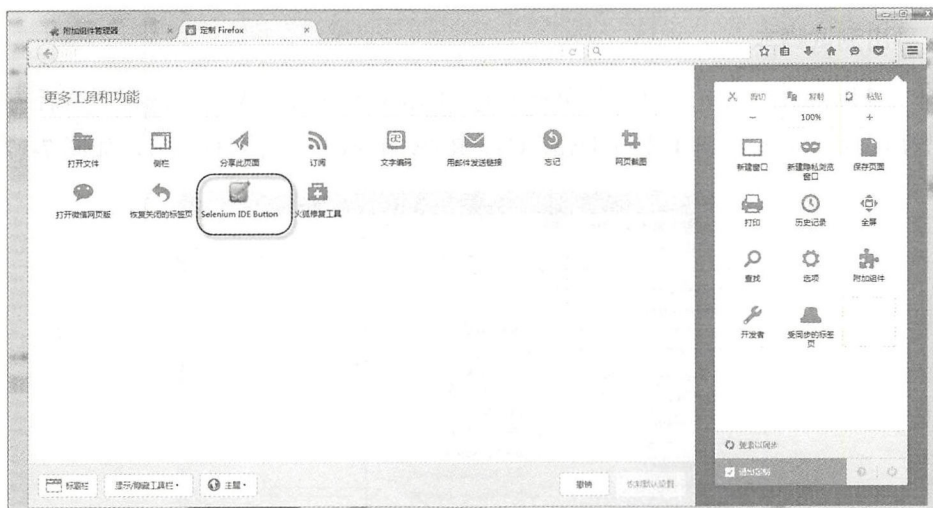



图 7-15 Selenium IDE 安装成功

2. 录制测试脚本

打开 SeleniumIDE，输入 URL，单击“录制脚本”按钮，如图 7-16 所示，会自动生成脚本代码。脚本是由一条一条的 Action(行为)组成，而每个 Action 又由 Command、Target

质量全面管控：从项目管理到容灾测试

和 Value 三者组成。

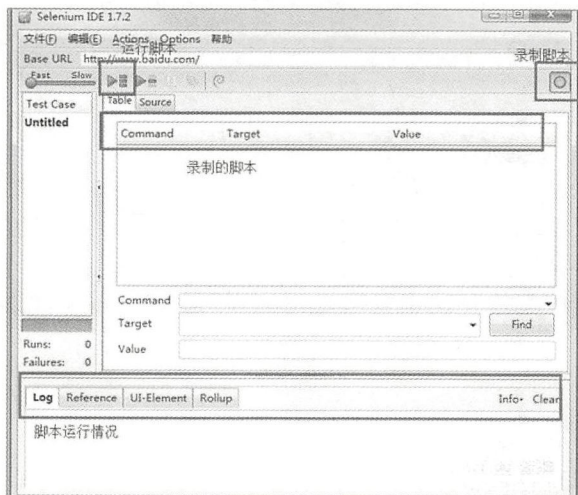


图 7-16 Selenium IDE 界面

3. 导出测试脚本

脚本录制完成后，单击“文件”菜单下的“Export Test Case As...”选项，然后选择相应语言，导出脚本。导出的脚本支持 Java、C#、Ruby 和 Python 等多种语言，如图 7-17 所示。

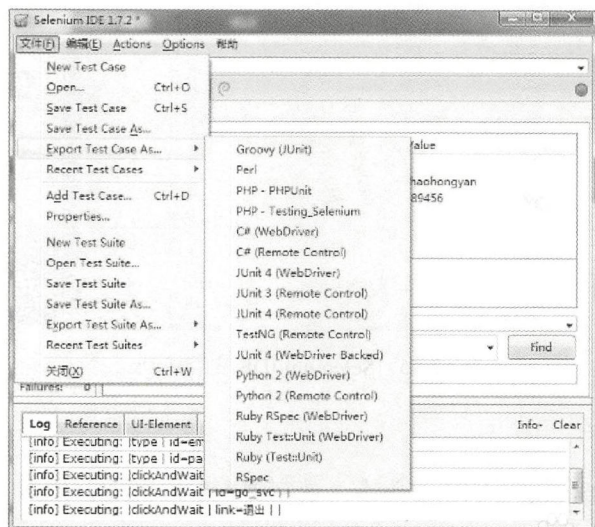


图 7-17 Selenium 导出测试脚本

7.2.3 Selenium 使用

Selenium 的脚本可以通过两种方式生成，第一种方式是使用 Selenium IDE 录制，但是这种方式录制的自动化脚本不够灵活，很难适应需求的变更，页面一旦发生变化就需要重新录制脚本，造成测试脚本维护成本较大。

另外一种方式是通过 WebDriver 中提供的 API 手工编写测试脚本，这种方式可以通过 C#、Python 和 Java 等编程语言实现，手工编写的脚本更具有面向对象和机构化的特性，脚本中的函数和方法能得到更好的复用，也比较易于维护和扩展。如果读者对开发语言比较熟悉，建议使用第二种方式编写测试脚本。下面介绍使用 Java 语言开发测试脚本的方法。

1. 脚本开发

(1) 开始使用 WebDriver 前，需要先安装 JDK，配置 Java 环境变量。然后下载最新版 Eclipse，下载网址为：<http://www.eclipse.org/downloads/>。

(2) 解压 Eclipse.zip 文件后，双击 Eclipse.exe 文件启动 Eclipse，在启动 Eclipse 过程中需要选择一个 Workspace 来保存测试项目，界面如图 7-18 所示。

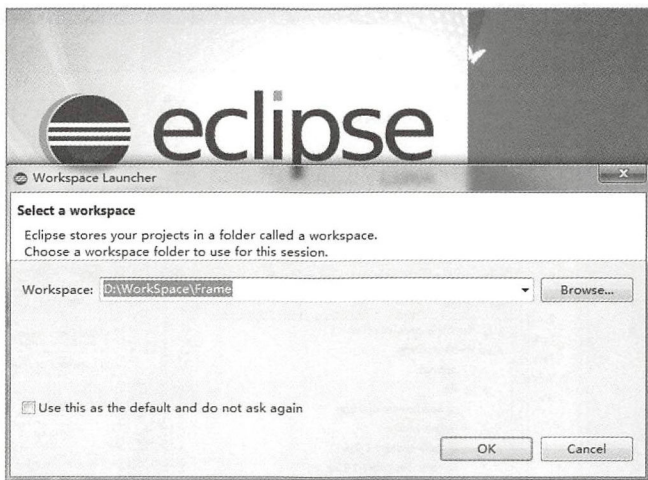


图 7-18 选择一个 Workspace

(3) 登录 Selenium 官网并下载 Selenium 的 jar 包，解压 selenium-java 压缩包后包含如图 7-19 所示的内容，本书使用的 Selenium 的版本为 2.39。

(4) 打开 Eclipse 后新建一个 Java Project，然后把解压后的 Jar 包复制到新建的 Project 目录下，目录结构如图 7-20 所示。

质量全面管控：从项目管理到容灾测试

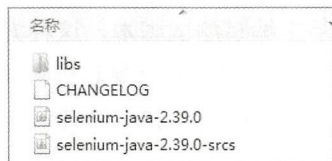


图 7-19 Selenium 相关 Jar 包

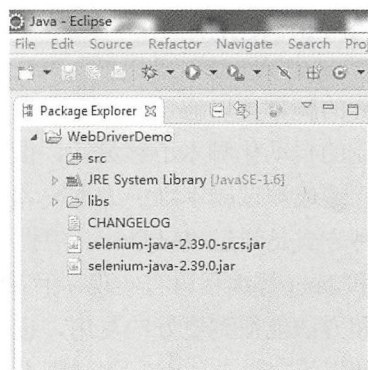


图 7-20 项目目录结构

(5) 添加 build path，项目目录右键选择“Build Path”选项，单击“Config build path”选项，出现 Java Build Path 弹出框，如图 7-21 所示。选择 Libraries 选项，单击“Add JARs”按钮，首先添加 libs 文件夹下的全部 Jar 包，然后添加 selenium-java-2.39.0.jar。如果需要查看 Selenium 源代码，可以添加 selenium-java-2.39.0-srcs.jar。

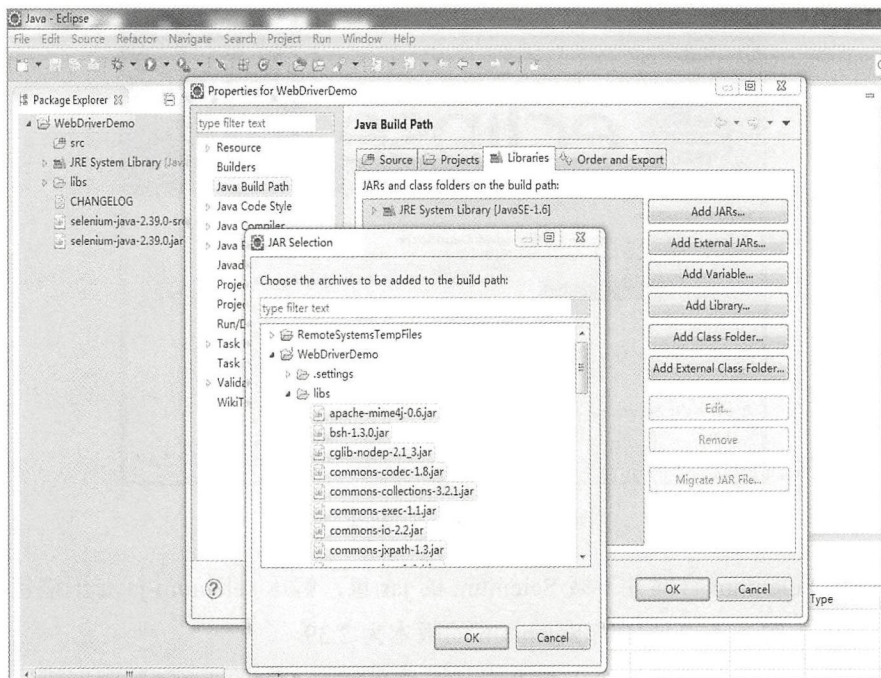


图 7-21 添加 Selenium Java Jar 包

(6) 添加后的结构中多了 Referenced Libraries, 这就是前面添加进去的 Jar 包, 如图 7-22 所示。

(7) 在 src 源代码目录下, 新建一个测试类, 如 TestHelloWorld, 如图 7-23 所示。

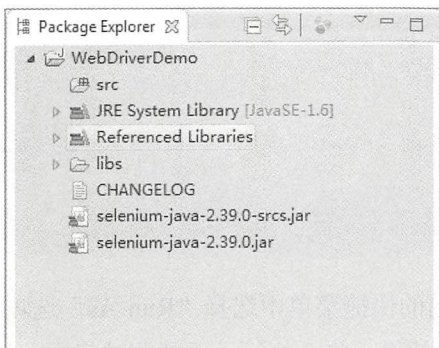


图 7-22 目录结构多了一项“Reference Libraries”

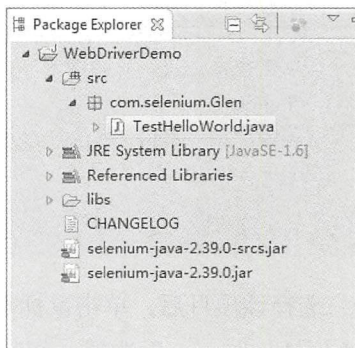


图 7-23 新建 Selenium 测试类

(8) 测试代码类的主要功能如下:

- 在 Firefox 浏览器中打开百度首页;
- 在搜索框输入 Test;
- 单击“搜索”按钮;
- 关闭浏览器。

```
package com.selenium.test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.*;
public class TestHelloWorld {
    public static void main(String[] args) {
        //如果 Firefox 浏览器没有默认安装在 C 盘, 需要制定其路径
        System.setProperty("webdriver.firefox.bin",
            "D:/firefox/firefox.exe");
        //创建一个 WebDriver 类
        WebDriver driver = new FirefoxDriver();
        //打开百度网址
        driver.get("http://www.baidu.com/");
        //最大化浏览器窗口
        driver.manage().window().maximize();
    }
}
```

```
//获得搜索框页面元素，根据 name 定位元素
WebElement txtbox = driver.findElement(By.name("wd"));
//搜索框中，输入“Test”值
txtbox.sendKeys("Test");
//获得搜索按钮的页面元素，根据 id 定位元素
WebElement btn = driver.findElement(By.id("su"));
//单击搜索按钮，开始搜索
btn.click();
//关闭浏览器窗口
driver.close();
}
```

(9) 选择该项目后，单击鼠标右键，在弹出的快捷菜单中选择“Run As”选项，然后单击“Java Application”选项，就可以看到运行效果，将会执行上面的测试步骤。

2. 页面元素查找

元素定位是自动化脚本的基础，只有先找到元素，才能对这个元素进行相关操作。在前面已经介绍了元素定位相关插件和部分插件的使用，本小节重点介绍通过 Selenium API 的相关方法来找到页面的某个元素所使用的技巧。

- 页面元素定位必须使用属性值的唯一属性，若页面中存在与其属性重复其他元素，则不能使用该属性定位；
- 优先使用 ID、Name、ClassName 和 LinkText 来查找元素；
- 如果使用 ID、Name、ClassName 和 LinkText 查找不到元素，可以使用 XPath、CSS 来定位元素。在特殊情况下可以通过 JavaScript 和 JQuery 来对元素定位；
- 定位元素必须使用固定的属性值，若该属性是动态生成的，每次运行的值均会发生变化，则不能使用该属性定位；
- 如果要定位的元素在 Iframe 中，需要先调用 SwitchTo().frame()方法进入该 frame，才能对 frame 中的元素进行操作。操作完 frame 中的元素后，需调用 SwitchTo().DefaultContent()方法释放，否则操作主页面时会无法找到元素。

Selenium API 中提供一个全局对象 By 来定位元素的位置。By 类中有许多属性，包含 ID、Name、XPath、ClassName 等。下面笔者通过实例来详细介绍运用 Selenium 中的 By 查找元素的方法。

(1) 使用 Firefox 浏览器打开百度首页, 如图 7-24 所示, 首页中包含一个文本框和一个按钮, 先定位文本框。



图 7-24 测试页

(2) 使用 Firebug 查看“文本框”页面代码, 如图 7-25 所示。



图 7-25 定位元素

(3) 通过表 7-4 中的 By 元素相关方法可以定位到“文本框”。

表 7-4 By 元素相关方法

方法名称	作用
By.id(“kw”)	使用 ID 查找其位置
By.name(“wd”)	使用 Name 查找其位置
By.classname(“s_ipt”)	使用 ClassName 查找其位置
By.xpath(“//input[@id='kw']”)	使用 XPath 查找其位置

(4) 把定位到的对象赋给一个 WebElement 对象, 可以对定位到的文本框进行操作, Java 代码如下:

```
WebDriver webdriver = new FirefoxDriver();
WebElement BaiDu=webdriver.findElement(By.id("kw"));
BaiDu.sendKeys("Hello World");
```


质量全面管控：从项目管理到容灾测试

(5) 在 B/S 系统中，可能会遇到前端框架使用 iframe 元素的情况，iframe 元素与普通元素定位稍有不同，定位 iframe 元素需要先进入到 iframe 里面，再操作该元素，操作完成后需要退出 iframe。例如 frame.html 内嵌 part1.html 和 part2.html，如图 7-26 和图 7-27 所示。

```
<html>
  <head>
    <title>Frame</title>
  </head>
  <body>
    <p id = "p">Outside frame</p>
    <iframe id = "f_1" src = "part1.htm"></iframe>
    <iframe id = "f_2" src = "part2.htm"></iframe>
  </body>
</html>
```

图 7-26 frame.html

```
<html>
  <head><title>Part1</title></head>
  <body>
    <p id = "f_p">This is part 1</p>
    <input id = "btn" type = "button" value = "click me" onclick = "alert('hello')" />
  </body>
</html>
```

图 7-27 part1.html

(6) 若要定位 id 为"f_p"的元素，由于元素位于一个 iframe 中，因此需要先进入到 id 为"f_1"的 frame 页面，才能操作元素，Java 代码如下：

```
WebElement iframeId=driver.findElement(By.name("f_1"));
driver.switchTo().frame(iframeId);
WebElement btnId= driver.findElement(By.id("btn"));
btnId.click();
```

(7) 操作完元素后，需要退出 iframe，返回主页面才能查找其他非 iframe 里的元素，调用如下方法：

```
driver.switchTo().defaultContent();
```

3. 相关 API 介绍

大部分 Selenium 的 API 可以通过 Selenium 官网查询到其使用方法，读者可以登录 Selenium 官网下载帮助文档，根据帮助文档编写脚本。表 7-5 列出了常用的 API 及其描述，仅供读者参考。

表 7-5 WebDriver 常用函数

API名称	函数描述	举例	举例描述
New FireFoxDriver()	实例化一个FireFox, 未启动FireFox	driver = New FireFoxDriver()	启动FireFox
New InternetExplorerDriver()	实例化一个IE, 未启动IE	driver = New InternetExplorerDriver()	启动IE
Get(String URL)	打开一个页面	driver.Get("www.baidu.com")	打开www.baidu.com
Navigate().to(String URL)	打开一个页面	driver.Navigate().to("www.baidu.com")	打开www.baidu.com
By.Id(String id)	通过ID查找该元素	By.Id("name")	查找ID为Name的元素
By.Name(string Name)	通过Name查找元素	By.Name("wd")	查找Name为Wd的元素
By.Xpath(string xpath)	通过Xpath查找元素	By.Xpath("//input[@id='name']")	通过xpath查找元素
By.LinkText(string link)	查找链接	By.LinkText("Link")	查找Link链接
FindElement(string Element)	查找对象	element=driver.FindElement(By.Id("name"))	找到ID为name的元素, 并转化为一个Element
sendKeys(string str)	在输入框里面输入字符	element.sendKeys("aaa")	向文本框中输入aa
Clear()	清空文本框内容	element.Clear()	清空文本框内容
GetText()	获取文本框内容	element.GetText()	获取文本框内容
Select()	实例化一个下拉选择框	element=driver.FindElement(By.Id("select")) Select select = new Select(element)	找到下拉选择框id为select的元素
SelectByVisibleText(string text)	选择选择项	select.selectByVisibleText("aa")	选择下拉项aa
SelectByValue(string text)	选择选择项	select.selectByValue("aa")	选择下拉项aa
GetAllSelectedOptions()	获取选择项的值	select.getAllSelectedOptions()	获取选择项值
Click()	点击一个对象	driver.findElement(By.id("BookMode")).click()	点击ID为BookMode的元素
isEnabled()	判断元素是否为disable	driver.findElement(By.id("save")).isEnabled()	判断ID为Save是否为disable
Submit()	提交一个表单	driver.findElement(By.id("approve")).submit()	提交表单
save_screenshot()	截图	webdriver.Firefox().save_screenshot("C:\error.jpg")	截图并保存到C:\error.jpg
switchTo()	Window和frame的切换	driver.switchTo().frame("leftFrame")	切换到leftFrame的Frame

7.3 接口自动化测试

随着互联网的发展和不同终端（Pad、Mobile、PC）的兴起，对开发人员的要求也越来越高，纯浏览器的应用已经不能完全满足用户体验和功能需求，往往需要针对不同的用户终端开发不同的版本。为了提升开发效率，前后端分离的需求越来越被重视，后端负责提供业务和数据接口，前端负责通过不同的终端来展现后端接口提供的数据。同一份数据接口，可以服务于 APP、H5、PC 端及第三方调用。后端测试在整个测试任务中越来越重要。后端测试主要在测试开发提供给前端接口的正确性，接口相比 PC 和 APP 的界面稳定，不会经常变动，接口自动化测试成为自动化测试主要切入点。

7.3.1 接口测试类型

在做接口测试前需要熟悉接口使用的协议，不同的协议其测试方法也不一样，现在大部分接口都采用 RESTful、Socket 和 Web Service 等方式实现，通过以下内容介绍这三种协议。

1. RESTful 架构

RESTful 架构是目前比较流行的一种互联网软件架构，即 Representational State Transfer 的缩写。从名字可以理解成资源在表现层的状态转换，主要包含如下几个方面。

- 资源：在 RESTful 中的资源应该是指表现层的资源。它是网络上的一个实体，或网络上的一个具体信息。它可以是一段文本、一张图片、一首歌曲、一种服务，总之就是一个具体的实体。你可以用一个 URI 指向资源，每种资源对应一个特定的 URI。要获取这个资源，访问它的 URI 就可以，因此 URI 就成了每一个资源的地址或独一无二的识别符；
- 表现层：“资源”是一种信息实体，URI 代表了资源的位置。表现层把“资源”以各种形式表现出来。比如文本可以用 txt 格式表现，也可以用 HTML 格式、XML 格式、JSON 格式表现，甚至可以采用二进制格式；
- 状态转换：访问一个网站，就代表了客户端和服务器的一个互动过程。在这个过程中，势必涉及到数据和状态的变化。RESTful 采用 HTTP 协议，HTTP 是一种无状态协议。可以使用 GET、POST、PUT、DELETE 让服务器端的状态发生变化。

2. Socket

Socket 通常也称为套接字，用 IP 地址和端口实现不同虚拟机或不同计算机之间的通信。Socket 连接包含以下 3 个步骤：服务器监听、客户端请求和连接确认。图 7-28 所示为 Socket 连接过程。

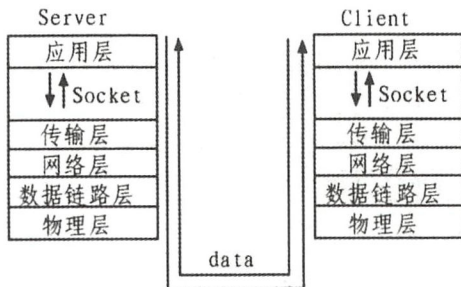


图 7-28 Socket 交互流程

- 服务器监听: 服务器开放一个端口, 一直处于监听客户端的请求, 实时监控网络状况。
- 客户端请求: 客户端套接字提出连接请求, 需要描述服务端套接字的地址和端口号, 然后向服务端套接字发送连接请求。
- 连接确认: 当服务器端套接字监听到或者说接收到客户端套接字的连接请求, 它就响应客户端套接字的请求, 建立一个新的线程, 把服务器端套接字的描述发给客户端。一旦客户端确认了此描述, 连接就建立好了。而服务器端套接字继续处于监听状态, 继续接收其他客户端套接字的连接请求。

3. Web Service

Web Service 是一种面向服务的技术架构, 通过标准的 Web 协议提供方法, 目的是保证不同平台数据的互操作性。Web 服务实际是一个软件系统, 用以支持不同机器的互相操作。它由大量的网络程序 API 组成, Web 服务器为这些 API 提供一个外界机器可以识别的服务, 执行客户所提交服务的请求。可以用任何你喜欢的语言编写 Web Service, 只要可以通过 Web Service 标准对这些服务进行查询和访问。如图 7-29 所示描述了 Web Service 的工作模型。

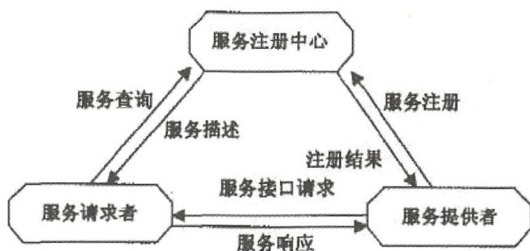


图 7-29 Web Service 工作模型

7.3.2 接口测试工具

接口测试是没有界面的测试, 需要开发一些测试工具和编写一些代码来辅助测试。市面上也有一些开源的接口测试工具, 例如 SoapUI、JMeter、Firefox 的 RestClient 插件和 Chrome 的 postman 插件, 它们都是非常出色的接口测试工具, 其中 SoapUI 和 JMeter 既可以做功能测试也可以做性能测试。在开始接口测试前, 应该先了解被测系统的接口所采用的协议是 HTTP、Socket 还是 Web Service, 才可以选择合适的工具对接口进行接口测试。如果没有现成的测试工具, 需要根据接口的协议来开发测试工具。

7.3.3 Mock 测试

软件的实现是充满依赖关系的，对于某些不容易构造或者不容易获取的对象，创建一个虚拟的对象进行测试的方法称为 Mock 测试，比如软件开发中 Service 层（服务层）主要负责业务模块的逻辑应用设计；DAO 层（数据库访问层）主要是做数据持久层的工作，负责与数据库进行联络和数据的增、删、改、查等操作。Service 层是建立在 DAO 层之上的，因而 Service 层依赖于 DAO 层的接口数据，代码的依赖性变大。如果想在不涉及依赖关系的情况下测试代码，无视代码的依赖关系保障测试代码的有效性，就产生了模拟对象的概念。创建一个可以替代实际对象的模拟对象，可以通过特定参数来调用特定方法，并且能返回预期结果。在模拟对象中，要关注三个方面：设置测试数据、设定预期结果和实际结果。以下代码示例为如何创建 Mock 的模拟对象。

(1) 新建一个 Person 实体类代码如下，包含用户 ID 和用户名称两个字段。

```
package com.bean;
public class Person {
    private final Integer personID;
    private final String personName;
    public Person( Integer personID, String personName )
    {
        this.personID = personID;
        this.personName = personName;
    }
    public Integer getPersonID()
    {
        return personID;
    }
    public String getPersonName()
    {
        return personName;
    }
}
```

(2) 编写 Person 类的数据库访问层代码如下，定义访问数据库的接口，在此不提供 DAO 的实现。

```
package com.dao;
import com.bean.Person;
public interface PersonDAO {
```

```

/*
 * 查找个人信息
 */
public Person find(Integer personID );
/*
 * 更新个人信息
 */
public void update(Person person );
/*
 * 保存个人信息
 */
public void save(Person person);
/*
 * 删除个人信息
 */
public void delete(Integer PersonID);
}

```

(3) 编写 Person 类的服务层代码如下，服务层通过调用数据库访问层的接口来实现。

```

package com.service;
import com.bean.Person;
import com.dao.PersonDAO;
public class PersonService {
    private final PersonDAO personDao;
    public PersonService( PersonDAO personDao )
    {
        this.personDao = personDao;
    }
    /**
     * 更新人员信息
     * @param personId
     * @param name
     * @return
     */
    public boolean update( Integer personId, String name )
    {
        Person person = personDao.find( personId );
        if( person != null )
        {
            Person updatedPerson = new Person( person.getPersonID(), name );
            personDao.update( updatedPerson );
        }
    }
}

```



```
return true;
    }
else
    {
return false;
    }
}
```

(4) 编写测试类测试 Person 服务层的代码如下，由于数据库访问层的接口没有具体实现，需要使用 Mock 来进行测试，以保证测试任务能顺利进行。

- 首先在测试启动 setUp() 时，实例化 Mock 对象：MockitoAnnotations.initMocks(this);
- 在实际测试方法 UpdatePerson_Test() 中，设置模拟数据库访问层的数据 Person(1, "Phillip"); 再使用模拟的 Person 数据来执行数据库访问层中的 update() 方法；最后使用参数捕获器 ArgumentCaptor 捕获 personDAO.update() 的传入参数；
- 设置断言，检测模拟数据和捕获参数是否一致。

```
public class PersonServiceMockTest {
    @Mock
    private PersonDAO personDAO;
    private PersonService personService;

    @BeforeMethod
    public void setUp()
        throws Exception
    {
        //实例化 Mock 对象
        MockitoAnnotations.initMocks(this);
        personService = new PersonService(personDAO);
    }

    @Test
    public void UpdatePerson_Test()
    {
        //创建 Bean 数据
        Person person = new Person(1, "Phillip");
        //模拟 DAO 数据
        when( personDAO.find(1)).thenReturn(person);
        //执行待测试的 service 方法
        boolean updated = personService.update( 1, "David" );
        //设置断言
    }
}
```

```
assertTrue(updated);
verify(personDAO).find(1);
    //用参数捕获器来捕获 Person 类传入方法的参数
    ArgumentCaptor<Person> personCaptor = ArgumentCaptor.forClass( Person.class );
    //捕获 personDAO.update 方法的参数
verify(personDAO).update(personCaptor.capture());
    //从捕获器中，获取传入方法的参数值
    Person updatedPerson = personCaptor.getValue();
    //设置断言
assertEquals( "David", updatedPerson.getPersonName() );
verifyNoMoreInteractions(personDAO);
}

@Test
public void UpdateIfPersonNotFound_Test()
{
    when(personDAO.find(1)).thenReturn(null);
    boolean updated = personService.update( 1, "David" );
    assertFalse(updated);
    verify(personDAO).find( 1 );
    verifyZeroInteractions(personDAO);
    verifyNoMoreInteractions(personDAO);
}
}
```

(5) 通过 TestNG（关于 TestNG 的相关知识请参考 7.4 小节）执行测试用例，测试报告显示如图 7-30 所示。报告中列出了测试方法 UpdatePerson 和 UpdateIfPersonNotFound 的结果。

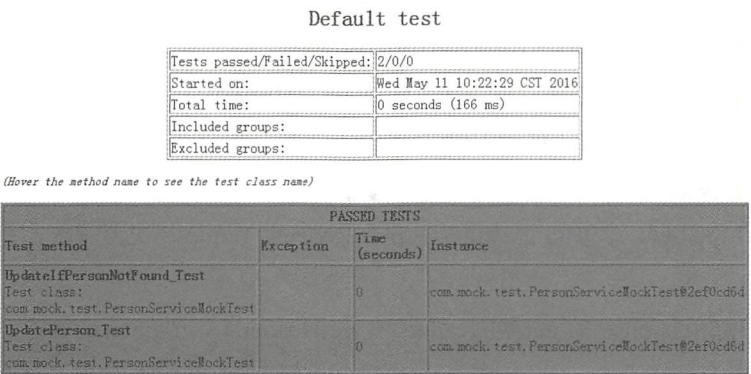


图 7-30 Mock 测试结果

7.3.4 HTTP 协议测试

HTTP 协议（HyperText Transfer Protocol，超文本传输协议）是因特网上应用最为广泛的一种网络传输协议，所有的 WWW 文件都必须遵守这个标准。HTTP 在接口测试中占有非常重要的位置。对于测试 HTTP 协议的接口，Apache 软件基金会组织开发了 HttpClient 项目，HttpClient 项目属于 Jakarta Common 下的子项目，提供了非常丰富的客户端编程工具包来支持 HTTP 协议，使客户端发送 HTTP 请求变得容易，方便了测试人员测试 HTTP 协议的接口。HttpClient 包括以下特性：

- 基于标准、纯净的 Java 语言。实现了 Http1.0 和 Http1.1，支持 HTTPS 和 HTTP 协议。
- 以可扩展的面向对象的结构实现了 HTTP 全部的方法（GET、POST、PUT、DELETE）。
- 通过 HTTP 代理建立透明的连接，自动处理 Cookie 中信息。
- Request 的输出流可以避免流中内容直接缓冲到 Socket 服务器。
- Response 的输入流可以有效地从 Socket 服务器直接读取相应内容。
- 直接获取服务器发送的 Response Code 和 Headers，并可以设置连接超时。
- 源代码基于 Apache License，可免费获取，方便进行二次开发。

比较著名的接口测试工具 JMeter 也采用了 HttpClient 包进行开发，通过它来发送和接收请求。下面将通过 HttpClient 中提供的方法来发送请求并接收响应，使用 HttpClient 一般步骤如下：

- ① 创建 HttpClient 对象。
- ② 创建请求方法的实例，并指定请求 URL。如果需要发送 GET 请求，创建 HttpGet 对象；如果需要发送 POST 请求，创建 HttpPost 对象。
- ③ 如果需要发送请求参数，可调用 HttpGet、HttpPost 共同的 setParams(HttpParams params)方法来添加请求参数；对于 HttpPost 对象而言，也可调用 setEntity(HttpEntity entity)方法来设置请求参数。
- ④ 调用 HttpClient 对象的 execute(HttpUriRequest request)发送请求，该方法返回一个 HttpResponse。
- ⑤ 调用 HttpResponse 的 getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用 HttpResponse 的 getEntity()方法可获取 HttpEntity 对象，该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

⑥ 释放连接。无论执行方法是否成功，都必须释放连接。

如下代码是通过 Apache 组织的 HttpClient 中的方法测试 Http 接口，该代码主要功能是：

- 通过 POST 方式添加部门信息；
- 通过 GET 方式获取部门信息；
- 检查添加部门信息是否成功；
- 检查获取部门信息是否正确。

(1) 新建一个 HTTP 的测试类发送 POST 请求，通过接口添加部门信息，部分代码如下：

```
/**
 * 发送 POST 请求添加部门信息
 */
@Test
public void Test_Post()
{
    String URL="http://127.0.0.1:8080/paiban/department/add";
    List<NameValuePair> formparams = new ArrayList<NameValuePair>();
    formparams.add(new BasicNameValuePair("describes", "测试管理中心"));
    formparams.add(new BasicNameValuePair("dname", "测试管理"));
    formparams.add(new BasicNameValuePair("deptno", "1"));
    // 创建默认的 httpClient 实例.
    CloseableHttpClient httpClient = HttpClient.createDefault();
    // 创建 httpPost 实例
    HttpPost httpPost = new HttpPost(URL);
    String Response=null;
    // 创建参数队列
    UrlEncodedFormEntity uefEntity;
    try {
        uefEntity = new UrlEncodedFormEntity(formparams, "UTF-8");
        httpPost.setEntity(uefEntity);
        System.out.println("executing request " + httpPost.getURI());
        //执行 execute () 发送请求 httpPost, 获取响应 response
        CloseableHttpResponse response = httpClient.execute(httpPost);

        try {
            HttpEntity entity = response.getEntity();
            // 检查响应状态
            AssertJUnit.assertEquals("HTTP/1.1 200 OK",response.getStatusLine());

            if (entity != null) {
                Response= EntityUtils.toString(entity, "UTF-8");
                AssertJUnit.assertEquals("{\"ok\":true}", Response);
            }
        }
    }
}
```




```
        }  
    } finally {  
        response.close();  
    }  
} catch (ClientProtocolException e) {  
    e.printStackTrace();  
}  
} catch (UnsupportedEncodingException e1) {  
    e1.printStackTrace();  
}  
} catch (IOException e) {  
    e.printStackTrace();  
}  
} finally {  
    // 关闭连接,释放资源  
try {  
        httpclient.close();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
}
```

(2) 发生 GET 请求获取部门信息，检查添加部门是否成功，是否能正确获取通过 POST 方式添加的部门信息，测试代码如下：

```
/**  
 * 发送 GET 请求获取部门信息  
 */  
@Test  
public void Test_Get()  
{  
    String URL="http://127.0.0.1:8080/paiban/department?dno=1";  
    CloseableHttpClient httpclient = HttpClients.createDefault();  
    try {  
        // 创建 HTTP 的 GET 请求  
        HttpGet httpget = new HttpGet(URL);  
        System.out.println("executing request " + httpget.getURI());  
        // 执行 GET 请求  
        CloseableHttpResponse response = httpclient.execute(httpget);  
    }  
    try {  
        // 获取响应实体  
        HttpEntity entity = response.getEntity();  
        // 检查响应状态
```



```
        AssertJUnit.assertEquals("HTTP/1.1 200 OK",response.getStatusLine());
    if (entity != null) {
        String ExtResponse="{\"deptno\":\"1\",\"dname\":\"测试管理中心\",\"describes\":\"测试管理\"}";
        AssertJUnit.assertEquals(ExtResponse,EntityUtils.toString(entity));
    }
    } finally {
        response.close();
    }
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        // 关闭连接，释放资源
    }
    try {
        httpclient.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    }
}
```

(3) 通过 TestNG 查看脚本运行的结果，如图 7-31 所示。

Default test			
Tests passed/Failed/Skipped:		2/0/0	
Started on:		Wed May 11 10:12:55 CST 2016	
Total time:		0 seconds (316 ms)	
Included groups:			
Excluded groups:			

(Hover the method name to see the test class name)

PASSED TESTS			
Test method	Exception	Time (seconds)	Instance
Test_Get		0	com.http.test.HttpTest@10e73e0
Test class: com.http.test.HttpTest			
Test_Post		0	com.http.test.HttpTest@10e73e0
Test class: com.http.test.HttpTest			

图 7-31 HTTP 协议测试结果



7.4 TestNG 框架

TestNG 是一个开源的测试框架，TestNG 表示 Test Next Generation（测试下一代）。其灵感来自 JUnit 和 NUnit 的，它优于 JUnit 尤其是采用了数据驱动，使测试数据和脚本做了分离，同时引入了配置文件并且不需要继承任何父类，消除了大部分的旧框架的限制，使开发人员能够编写更加灵活强大的测试。因为它在很大程度上借鉴了 Java 注解（JDK5.0 引入的）来定义测试，可以用于集成测试。这些特性使得大多数自动化测试选择 TestNG，主要表现如下：

- 支持注解和 XML 使编写测试脚本更加方便
- TestNG 使用 Java 和面向对象的功能
- 支持综合类测试
- 支持数据驱动测试
- 支持依赖测试方法，并行测试，负载测试
- 灵活的插件 API
- 支持多线程测试。

7.4.1 TestNG 配置

TestNG 用一个插件集成在 Eclipse 中，由于 Eclipse 开发工具本身不自带该插件，如果要使用 TestNG，需要手动安装该插件，安装插件的方法如下：

（1）打开 Eclipse，选择菜单 Help 单击 Install New Software 选项，然后单击 Add 按钮，添加 Location 为“<http://beust.com/eclipse>”，如图 7-32 所示。

（2）对于不能上网的添加 Eclipse 插件的用户可以下载 TestNG 插件并解压，解压后有两个文件夹 features 和 plugins，如图 7-33 所示。

（3）将 features 下的文件夹复制到 Eclipse 安装目录的 features 中。

（4）将 plugins 下的文件夹复制到 Eclipse 安装目录的 plugins 中。

（5）打开 Eclipse，选择菜单命令“file”→“new”→“other”，项目类型中出现 TestNG 界面，如图 7-34 所示，表示 TestNG 安装成功。



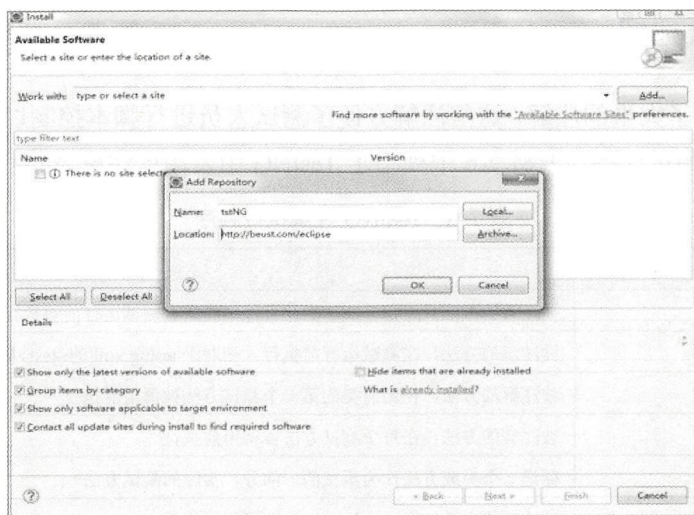


图 7-32 添加 TestNG 插件

名称	修改日期	类型	大小
features	2012/11/12 9:07	文件夹	
plugins	2012/11/12 9:07	文件夹	

图 7-33 TestNG 插件目录

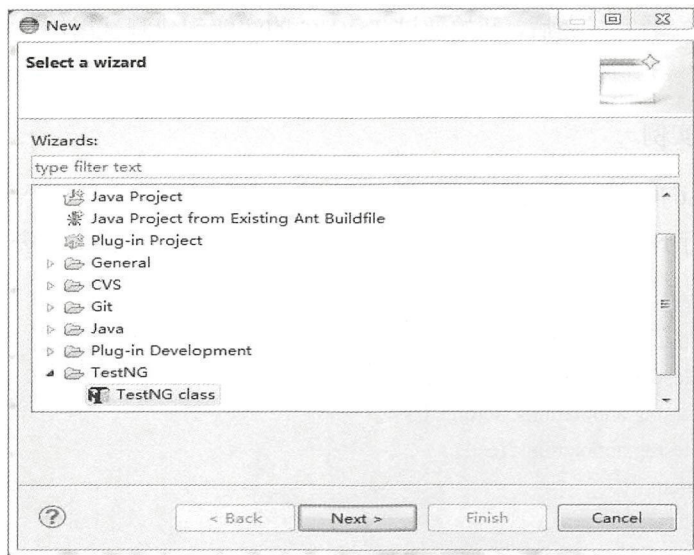


图 7-34 TestNG 安装成功





7.4.2 TestNG 注解

TestNG 提供了大量的注解，这些注解方便了测试人员进行脚本控制以及设置测试执行顺序，从而完成测试任务。表格 7-6 中列举了 TestNG 中常用的注解及执行的顺序。

表 7-6 TestNG 注解执行顺序

注解名称	描述
@BeforeSuite	被注解的方法只运行一次，在此套件中的所有测试运行前被执行
@BeforeTest	被注解的方法，在测试运行前执行，相对于 testng.xml 的 <test> 标签
@BeforeClass	被注解的方法，在当前类的第一个测试方法被调用前执行
@BeforeMethod	被注解的方法，在每个测试方法被调用前执行
@Test	标记一个类或方法作为测试的一部分，实际的测试方法
@AfterMethod	被注解的方法，在每个测试方法被调用后执行
@AfterClass	被注解的方法，在当前类的所有测试方法被调用后执行
@AfterTest	被注解的方法，在测试运行后执行
@AfterSuite	被注解的方法只运行一次，此套件中的所有测试运行之后被执行

7.4.3 测试套件

TestNG 插件安装完成之后，开始通过 TestNG 的注解创建测试用例，并用 TestNG 来执行这些测试用例。

1. TestNG 实例

首先使用 TestNG 创建并执行测试用例，编写一个测试类，定义一个测试方法 test，这个方法执行前执行 beforeClass 方法，以及运行结束后执行 afterClass 方法，代码如下：

```
package com.testng;
import org.testng.AssertJUnit;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
public class demo {
    @BeforeClass
    public void beforeClass()
```





```
{
    System.out.print("beforeclass");
}
@Test
public void test()
{
    AssertJUnit.assertEquals("Test", "Test");
    System.out.print("beforeclass");
}
@AfterClass
public void afterClass()
{
    System.out.print("afterClass");
}
}
```

2. 运行方式

运行 TestNG 编写的测试方法有 3 种，具体如下。

(1) 在要执行的方法上单击鼠标右键，在弹出的快捷菜单中选择“Run As”选项后，选择“TestNG Test”选项，如图 7-35 所示，开始执行测试脚本。

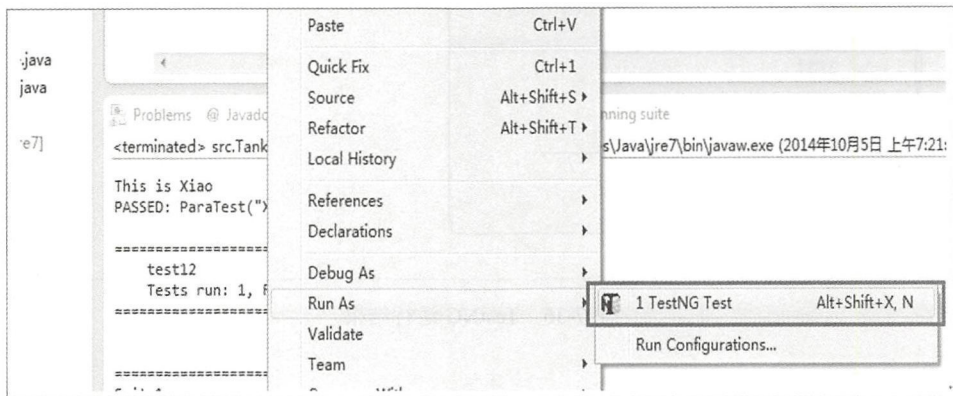


图 7-35 通过菜单运行测试脚本

(2) 通过 testng.xml 文件来运行，通过如下代码把要执行的 Case 名称放入 testng.xml 文件中，在 testng.xml 上单击鼠标右键，在弹出的快捷菜单中选择“Run As”选项，开始运行测试脚本。





```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite guice-stage="DEVELOPMENT" name="Default suite">
  <test verbose="2" name="Default test">
    <classes>
      <class name="com.testng.demo"/>
    </classes>
  </test>
</suite>
```

测试脚本也可以通过命令行来运行，命令如下：

```
java org.testng.TestNG testng.xml
```

(3) 运行完测试脚本以后，在项目的目录下会生成一个 test-output 的文件夹，里面存放着测试结果。找到 index.html，通过浏览器可以查看测试结果，如图 7-36 所示。

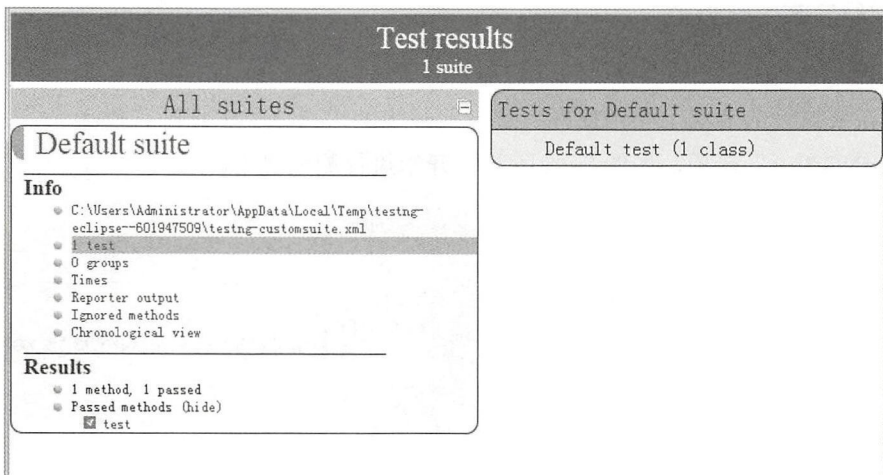


图 7-36 TestNG 运行结果

3. 顺序执行

在 testng.xml 中可以控制测试用例的执行顺序。当 test 元素属性 preserve-order="true" 时，可以保证节点下面的所有方法都是按顺序执行的，实例代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
```





```
<suite name="test">
  <test name="test01" preserve-order="true">
    <classes>
      <class name="com.testng.demo">
        <methods>
          <include name="FirestTest"></include>
          <include name="SecondeTest"></include>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

4. 组测试

在 TestNG 中，组测试是一个新的功能，它允许对测试方法进行分组，不仅可以声明属于群体的那些方法，并且可以指定一组包含其他组。TestNG 可调用和要求包括一组特定的群体，而排除另一个集合。这些功能提供了极大的灵活性，如果想运行两套不同的测试方法，则只需把不同的测试方法分成两个组，而不要求重新编译任何东西。分组时，需要在注解 Test 后面添加 groups = {"groupname"}，代码示例如下：

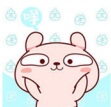
```
import org.testng.annotations.Test;

public class GroupTest {

    @Test(groups={"Test_A"})
    public void Group1()
    {
        System.out.println("Group_1");
    }

    @Test(groups={"Test_B"})
    public void Group2()
    {
        System.out.println("Group_2");
    }

    @Test(groups={"Test_A"})
```





```
public void Group3()
{
    System.out.println("Group_3");
}

@Test(groups={"Test_A","Test_B"})
public void Group4()
{
    System.out.println("Group_4");
}
}
```

5. 忽略测试

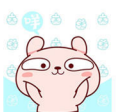
有时某个测试用例的测试代码还没有准备好，或者已经明确知晓被测系统的该功能有问题，这时如果还将此测试用例写入到测试方法，那么代码将无法运行。在这种情况下，`@Test(enabled = false)`有助于禁用此测试用例。如下代码中的 `TestDisable` 测试方法标注了 `@Test(enabled = false)`，那么该未准备好测试的测试用例将会被绕过，不被执行。

```
import org.testng.annotations.Test;

public class GroupTest {

    @Test(enabled=false)
    public void TestDisable()
    {
        System.out.print("这个脚本不允许");
    }

    @Test
    public void Demo()
    {
        System.out.print("test");
    }
}
```





6. 依赖测试

有时候脚本 B 需要依赖脚本 A 的运行结果，就需要按照一个特定的顺序调用方法，或有一个方法先初始化测试数据。TestNG 支持这种测试方法之间的显式依赖。TestNG 支持声明，也允许指定依赖。如下测试代码示例描述了只有等待 Setup 的方法通过，Message 才能运行。这种方式和顺序执行的最大差异是：依赖测试是 Setup 执行成功后，才会执行 Message 的方法；而顺序则不管 Setup 测试结果如何，都会执行 Message 测试方法。

```
import org.testng.annotations.Test;

public class Test {

    @Test
    public void Setup()
    {
        System.out.print("开始执行前准备");
    }

    @Test(dependsOnMethods={"setup"})
    public void Message()
    {
        System.out.print("Setup 方法执行失败这个方法不执行");
    }

    @Test
    public void Test()
    {
        System.out.print("我会一直执行下去");
    }
}
```

7.4.4 数据驱动

TestNG 的另一个功能是数据驱动。在大多数情况下，会遇到这样一个场景，业务逻辑需要不同测试数据进行测试。TestNG 提供的数据驱动模式允许开发人员重复运行同样的测试步骤，一遍又一遍，却可以使用不同的测试数据。通过 TestNG 实现数据驱动方式如下。



质量全面管控：从项目管理到容灾测试

(1) 使用 TestNG 的 XML 文件, 在 xml 里定义 `<parameter name="myname" value="test001">` 代码如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="test">
  <test verbose="2" name="test">
    <parameter name="myname" value="test001"></parameter>
    <classes>
      <class name="com.demo.test" />
    </classes>
  </test>
</suite>
```

(2) 添加测试类 Test, 通过注解 Parameters 获取定义在 XML 中的参数, 代码如下:

```
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class Test {

    @Test
    @Parameters("myname")
    public void ParaMeterTest(String myname)
    {
        System.out.println("参数值是:" + myname);
    }
}
```

(3) 在 TestNG 中还有一种方式, 通过 DataProvider 给测试类注入测试数据, 如下代码是在 Test 方法中添加注解 DataProvider="user", 其中 user 就是需要使用的参数源, 数据源是通过 user 方法提供的。

```
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class Test {

    @DataProvider(name="user")
```



```

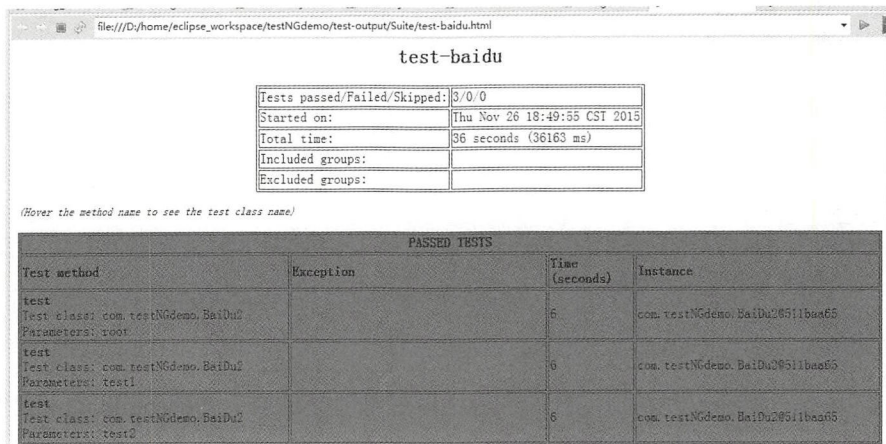
public Object[][] user()
{
    return new Object[][] {
        {"test01", "12345"},
        {"test02", "abcd"},
        {"test03", "YYYYY"}
    };
}

@Test(dataProvider="user")
public void test(String username, String passwd)
{
    System.out.print("用户名:"+username+"密码:"+passwd);
}
}

```

7.4.5 执行测试结果

测试报告是测试执行时最重要的输出部分，它可以帮助用户了解执行测试、故障点和失败原因的结果记录。另一方面，重要的是要留意执行流程，或在任何故障的情况下进行调试和分析。默认情况下，TestNG 会产生不同类型的测试执行报告，包括 HTML 和 XML 报表输出。测试报告位于"test-output"目录下，执行结果如图 7-37 所示。



The screenshot shows a web browser displaying a TestNG HTML report. The title is 'test-baidu'. Below the title is a summary table with the following data:

Tests passed/Failed/Skipped:	3/0/0
Started on:	Thu Nov 26 18:49:55 CST 2015
Total time:	36 seconds (36163 ms)
Included groups:	
Excluded groups:	

Below the summary table is a note: "(Hover the method name to see the test class name)".

The main section of the report is titled 'PASSED TESTS' and contains a table with the following data:

Test method	Exception	Time (seconds)	Instance
test Test class: com.testng.demo.Baidu2 Parameters: root		6	com.testng.demo.Baidu2@511ba6f5
test Test class: com.testng.demo.Baidu2 Parameters: test1		6	com.testng.demo.Baidu2@511ba6f5
test Test class: com.testng.demo.Baidu2 Parameters: test2		6	com.testng.demo.Baidu2@511ba6f5

图 7-37 TestNG 执行结果

质量全面管控：从项目管理到容灾测试

7.4.6 测试集成

在开发脚本中，读者可能会用 ANT 或者 Maven 来构建脚本，TestNG 能和这些工具集成，可以通过 ANT 或者 Maven 来运行 TestNG 编写的测试脚本。下面示例通过 ANT 来构建 TestNG 的测试脚本。

(1) 如果使用 Eclipse 开发脚本，默认自带 ANT 插件，无须下载。

(2) 新建 Java 项目，并创建 src、test 和 lib 三个文件夹。在 src 文件夹下新建要测试的 Java 类 Calc，代码如下。

```
public class Calc {
    //两个数相加
    public int sum(int a, int b){
        return a+b;
    }
    //两个数相减
    public int sub(int a, int b){
        return a-b;
    }
}
```

(3) 在 test 文件夹下新建测试类 TestCalc，测试类定义了两个测试方法 testsum 和 testsub 来计算加法和减法。

```
public class TestCalc {
    Calc calc = new Calc();
    @Test
    public void testsum() {
        Assert.assertEquals(5,calc.sum(2,3));
    }
    @Test
    public void testsub() {
        Assert.assertEquals(1,calc.sub(3,2));
    }
}
```

(4) 把 testng-6.8.jar 放入到 lib 文件夹，并在项目根目录下新建 build.xml 文件，用来执行 ANT 任务，build.xml 文件代码如下。

```
<project name="TestNG" default="test" basedir=".">
```

```
<taskdef name="testng" classname="org.testng.TestNGAntTask">
<classpath>
<pathelement location="lib/testng-6.8.jar"/>
</classpath>
</taskdef>

<property name="testdir" location="test" />
<property name="srcdir" location="src" />
<property name="libdir" location="lib" />
<property name="full-compile" value="true" />
<path id="classpath.base"/>
<path id="classpath.test">

<fileset dir="${libdir}">
<include name="**/*.jar" />
</fileset>

<pathelement location="${testdir}" />
<pathelement location="${srcdir}" />

<path refid="classpath.base" />
</path>

<target name="clean" >
<delete verbose="${full-compile}">
<fileset dir="${testdir}" includes="**/*.class" />
</delete>
</target>

<target name="compile" depends="clean">
<javac srcdir="${srcdir}" destdir="${testdir}" verbose="${full-compile}">
<classpath refid="classpath.test"/>
</javac>
</target>

<target name="test" depends="compile">
<testng outputdir="${testdir}" classpathref="classpath.test">
<xmlfileset dir="${srcdir}" includes="testng.xml"/>
</testng>
</target>
</project>
```


质量全面管控：从项目管理到容灾测试

(5) 右击 build.xml，选择“run as”选项并单击“ant build”选项来运行测试脚本，测试结果如图 7-38 所示。可以看到本例中，执行了 2 个测试，0 个测试失败，0 个测试跳过执行。

```
test:
[testng] [TestNG] Running:
[testng]   C:\TestNG_WORKSPACE\Test\src\testng.xml
[testng]
[testng] =====
[testng] Plug ANT test Suite
[testng] Total tests run: 2, Failures: 0, Skips: 0
[testng] =====
[testng]

BUILD SUCCESSFUL
Total time: 1 second
```

图 7-38 通过 ANT 执行结果

7.5 要点回顾

通过对本章的学习，读者应该了解了自动化测试流程和特点。根据被测系统不同，选用的自动化测试工具也不同，比如 QTP、Selenium 擅长 Web 方面自动化测试，而 HttpClient 主要用于接口测试中。

通过 Selenium 实施 Web 自动化测试的时候，需要先对页面上的元素进行定位，找到元素以后，执行相关操作；运用 Selenium API 来编写 Web 测试自动化脚本，通过预期结果和实际结果的比较来判断测试结果的正确性；编写测试用例时候可以通过 TestNG 提供的数据驱动，实现数据和脚本分离；采用 TestNG 执行测试用例，生成测试报告，测试报告结果可以通过 HTML 和 XML 来展示；测试工程师可以根据测试报告进行分析，根据测试报告分析是测试脚本问题还是系统本身的缺陷，如果发现系统缺陷及时通知相关人员进行修复。

对应接口自动化测试，首先需要了解接口所使用的协议，比如 HTTP、Web Service 和 Socket 等，通过这些协议编写自动化测试脚本，在接口测试中使用 HttpClient 来编写 HTTP 协议的测试脚本。自动化脚本是一个不断完善的过程，在测试前期测试脚本不稳定，需要后期通过不断的调试优化来提高测试脚本的健壮性。

第 8 章

自动化测试框架

通过第 7 章的介绍，相信读者已经熟悉了自动化测试的相关工具，以及如何编写自动化脚本，并使用 TestNG 框架执行测试用例，输出测试结果。在本章中，我们将通过一个真实的项目了解自动化测试框架，开发该框架的背景主要是为了实现公司“以客户需求为导向”的战略转型，这必然对业务流程优化再造和服务模式创新提出更多、更高的业务需求。公司积极快速响应市场需求、适应市场变化，必然在项目上提出更多、更紧急的业务需求。项目迭代次数增加，测试的需求也会呈爆发式增长，上线后风险不能把控，因此急需一种测试方法来满足这种爆发式的需求增长。自动化测试框架理念应运而生，希望在保持原有的人力资源基础上，提高现有的测试质量和效率。

本章内容将详细讨论自动化测试框架的思想，采用业务流程的分析方法进行框架的设计和分析，把手工用例如何转换成自动化用例，使用 Java 语言自主开发测试框架、编写测试脚本，使用流行的 Git 版本控制工具维护管理测试脚本，采用 Jenkins 做持续集成，通过定时任务来运行测试脚本，搭建分布式测试环境在最短的时间完成测试任务，最后通过邮件发送测试报告。本章主要内容如下：

- 测试框架分析
- 测试框架设计
- 测试框架开发
- 脚本开发
- 持续集成

8.1 框架分析

自动化测试框架提供一整套自动化测试 workflow。采用业务流程分析方法，先熟悉业务系统，然后根据业务系统划分出业务流程。业务流程指系统的一个完整的流程，例如用户购物流程，首先是用户登录系统，然后选择日期和价格以及需要购买的产品后，提交订单生成订单号，最后付款。这样一个完整的流程称为一个业务流程。这个业务流程可以拆分成相互独立的功能模块，比如用户登录、产品购买、提交订单、付款，这些功能模块也称为业务组件。由不同的业务功能模块就可以组成不同的业务流，而业务流程的实现可以定制化，这就是业务流程分析方法的精髓。通过下面的知识将进一步了解业务流和业务组件在业务流程分析方法中的重要作用。

8.1.1 框架设计目标

有效地使用自动化测试框架，不仅可以提高回归测试的执行效率和优化人力资源，而且便于满足业务需求变更和数据变更，便于不同的用户使用自动化测试框架。自动化框架把提供业务流可维护、数据可定制、脚本组件化以及数据和脚本相分离为目标。建立一套自动化测试框架相关的规范和管理制度，从而形成一套完备的自动化测试框架，主要体现在以下几个方面：

- 自动化测试框架将实现对测试资产库的统一管理，测试资产库包括：测试需求、测试计划、测试用例、测试数据、测试脚本、测试缺陷、测试日志和测试报告。自动化测试框架使测试资产库得到合理有效地复用，从而快速复用于回归测试；
- 开发管理测试脚本，定时运行测试脚本发送测试报告；
- 通过自动化测试框架，实现脚本和数据分离；
- 框架通过自主定制开发，方便扩展，易于维护；
- 引入“业务流程”分析方法，梳理自动化测试用例；
- 通过自动化测试框架，部分功能测试通过自动化测试来实现，提高测试效率。

8.1.2 业务流程层次分析

一个业务流程可分为多个业务功能模块，一个业务功能模块又可细分成多个业务单元。

每个业务单元可以执行不同的操作，如图 8-1 所示为业务流程分析过程。一般将业务功能模块与业务单元统称为业务子流程。整个业务流程按层次分解成业务流程、业务子流程和执行操作。

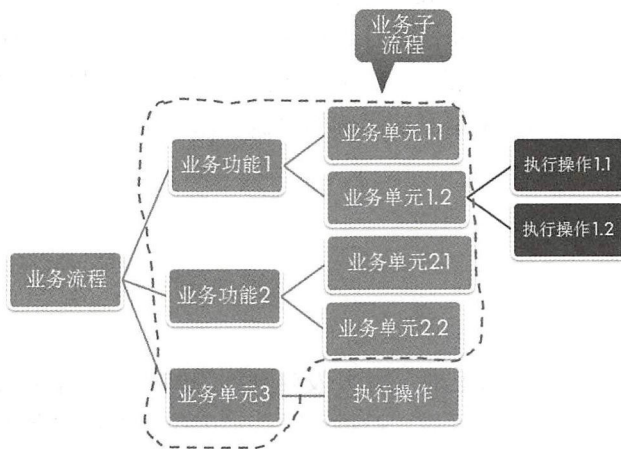


图 8-1 业务流程层次分析

8.1.3 业务流程测试自动化

通过业务流程分析可将业务流程分解成多个业务组件，多个业务流程中可能包含相同业务组件，比如用户登录、退出等公共模块，对业务流程进行组件化开发后，可降低开发与维护的工作量。一个业务组件变化，不需要修改其他组件，功能测试人员只需要根据业务规则设计测试用例关注业务，而自动化测试人员根据测试用例编写测试脚本关注技术的实现，实现业务与技术的分离。设计业务流程组件的原则如下。

- 业务功能的复用性。

当某个业务功能同时被多个业务流程所覆盖时，建议组件化。

- 业务功能的复杂度。

当某个业务功能本身的业务逻辑与前台界面较为复杂时，建议组件化。

- 业务功能的独立性。

当某个业务功能可以完成一个独立功能时，建议组件化。

- 业务功能的模块化。

当某个业务功能联系比较紧密时，可以使其模块化，能更好维护该模块。

质量全面管控：从项目管理到容灾测试

8.1.4 手工用例自动化

自动化测试用例的设计通常存在两种情况，第一种情况是已有手工用例库的老系统，第二种情况是无手工用例库的新系统。自动化测试可以发生在系统集成测试阶段，但更多体现在回归测试阶段。在针对某系统进行自动化测试覆盖时，采用“业务流程分析法”设计业务流程用例，针对无用例库的新系统可以直接设计业务流程自动化测试用例，对于已有手工用例库的老系统，则可采用测试用例的“拆并原则”对手工用例进行转换，形成自动化测试用例。

8.2 框架设计

自动化测试框架的建设主要包括设计开发自动化测试核心支撑框架，建立一套自动化测试脚本设计、脚本开发、扩展开发、数据维护、调度执行的模式和规范，保证自动化测试实施技术和方法的一致性，使自动化测试过程中各种角色和工作环节能够有效连接和运转。自动化测试框架由自动化测试工具和自动化测试框架两部分构成。

自动化测试工具包含如下内容：

- 被测系统属于 B/S 架构，可以使用 Selenium 负责页面对象识别操作；
- ANT 用于测试脚本的构建；
- Eclipse 负责脚本开发和调试工具；
- TestNG 负责测试脚本运行和报表输出。

自动化测试框架有如下功能：

- 自动化用例的编写和导入，手动用例和自动化脚本相关联；
- 为测试脚本开发提供统一的功能接口；
- 完成业务流程定义、业务组件定义、测试数据维护和运行参数设置；
- 管理测试脚本和执行机的调度运行；
- 测试结果的统计、输出和展示。

8.2.1 框架设计思想

在传统的自动化测试方式下，开发自动化脚本难度大，脚本维护成本高，对于测试资产库缺乏统一的管理，重要的测试资产库无积累、无共享、无复用，需要自动化测试人员

掌握一定的开发技能，不利于大规模的普及使用。自动化测试框架的引入将业务与技术相分离，不同的测试人员可以互相协作完成自动化用例的开发和维护，注重测试资产库的积累和复用，提高测试的效率。

自动化测试框架以业务与技术相分离、脚本与数据相分离、统一的自动化工作模式和测试资产库的不断积累为设计思想。如图 8-2 所示为自动化框架设计思想的抽象。

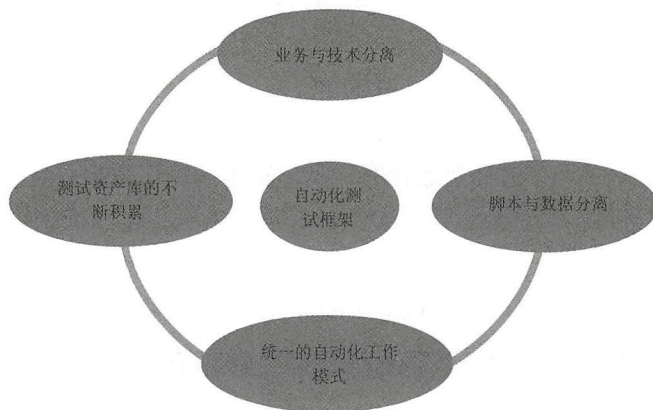


图 8-2 自动化测试框架的设计思想

- 业务与技术分离

在自动化框架的设计过程中，首先实现被测系统的业务逻辑和实现这些逻辑的脚本相分离。框架提供协同工作平台，由功能测试工程师设计业务组件和业务流，由自动化测试技术人员完成具体自动化脚本的开发，两个角色分工明确、高效配合。

- 脚本与数据分离

实现测试脚本和测试数据相分离，脚本和数据分别独立构建，使得同一套测试脚本可以使用不同的测试数据，降低脚本和数据的维护。基于以上的设计思路，自动化测试框架可以使用较低成本满足高复用、快速开发的要求。

- 统一的自动化工作模式

自动化测试框架固定了一套自动化脚本设计、自动化脚本开发、扩展开发和调度执行的模式和规范，保证了自动化测试技术和方法的一致性。

- 测试资产库的不断积累

自动化测试框架注重自动化测试资产的积累、共享和复用。自动化测试过程中，对产生的业务组件、业务流、测试脚本、测试数据等资产库进行有效的管理，并在不同的测试中实现复用。

8.2.2 框架物理架构

开发人员每天完成代码开发后将代码提交到 Git 服务器，Jenkins 服务器会自动从 Git 服务器中提取最新的代码，定时完成每日集成的任务，集成完毕后通过调用测试执行机启动测试人员已经编写好的测试脚本进行定时的回归测试，并用邮件将生成的集成结果和测试结果发送到相关人员的邮箱。如图 8-3 所示为自动化执行的流程，从物理上进行划分，应包含如下资源：

- 开发人员：负责代码的编写、调试和提交。
- 测试人员：负责测试脚本的编写、调试和提交。
- Git 服务器：用于存放框架运行库、测试脚本和测试数据。
- Jenkins 服务器：用于设置运行参数、管理测试执行机、控制执行测试、收集测试结果。
- 测试执行机：执行自动化测试脚本的计算机。

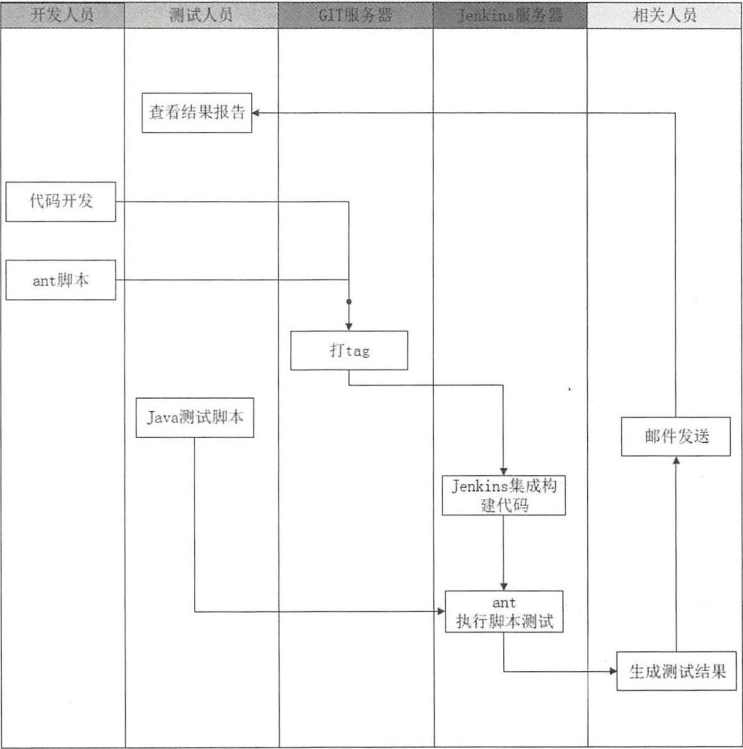


图 8-3 自动化执行流程

8.2.3 框架逻辑架构

自动化框架其实也是一套系统，它需要进行逻辑分层，以此满足平台的扩展性和可维护性，适应后期的自动化平台维护，如图 8-4 所示是本套自动化测试框架的逻辑架构，由定义层、测试引擎驱动层和被测系统层组成。

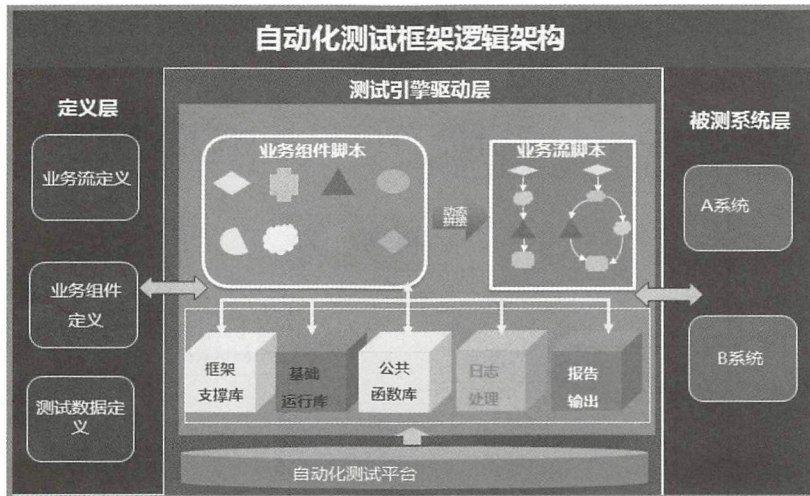


图 8-4 自动化测试框架的逻辑架构

1. 定义层

定义层包括业务流定义、业务组件定义和测试数据定义。其中，业务组件是系统中一个独立的功能模块，业务流定义系统运行的流程；测试数据定义是完成被测项目中输入数据类型定义、业务流中数据依赖关系以及测试过程中输出的数据。

2. 测试引擎驱动层

测试引擎驱动层是自动化测试框架的核心，根据业务流配置自动加载对象库、函数库和业务组件脚本以形成业务流脚本，并驱动脚本执行。通过驱动层实现了测试中业务和技术的分离。

3. 被测系统层

被测系统层支持对多浏览器、C/S 架构及基于移动设备上的 B/S 架构系统的自动化测试。

8.2.4 框架工作流程

自动化测试框架使功能测试工程师和自动化测试工程师能够在平台协同办公。功能测试工程师登录测试框架编写测试用例，填写测试数据，配置业务流程和业务组件；自动化工程师根据测试用例编写调试脚本，上传测试脚本和关联测试用例；自动化测试框架根据定义好的执行策略执行测试并生成报告，然后发生报告给相关人员；最后根据测试报告分析测试结果，得出系统是否通过本次自动化测试的结论。如图 8-5 所示描述了自动化框架的工作流程。

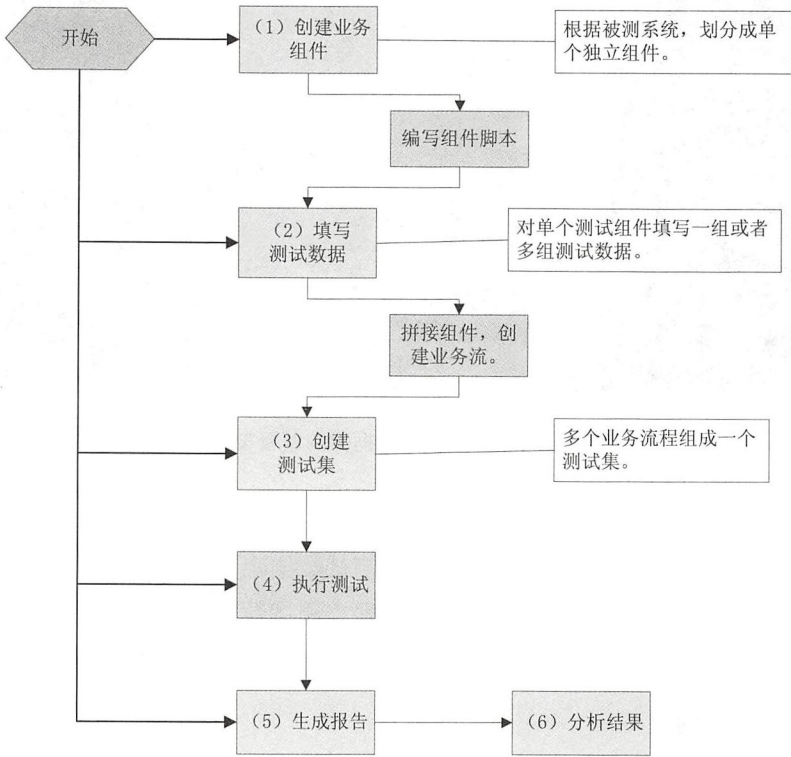


图 8-5 自动化框架的工作流程

8.3 框架开发

在开发测试框架的时候需要考虑测试用例如何维护，测试数据采用 Excel 还是 DB 来

进行存储，还需要提供公共方法给测试工程师以简化脚本的编写。如果脚本运行出错，可以通过日志分析其原因。测试脚本运行结束需要生成测试结果报告，通过测试报告了解测试覆盖率。本节将通过以上几个方面来介绍自动化测试框架。

8.3.1 创建测试用例

通过平台创建测试用例，可以结合 TestLink 或 TestRail 做二次开发形成测试框架的用例管理。下面使用笔者自主研发的自动化平台作为测试用例，图 8-6 所示描述了创建测试用例的过程。

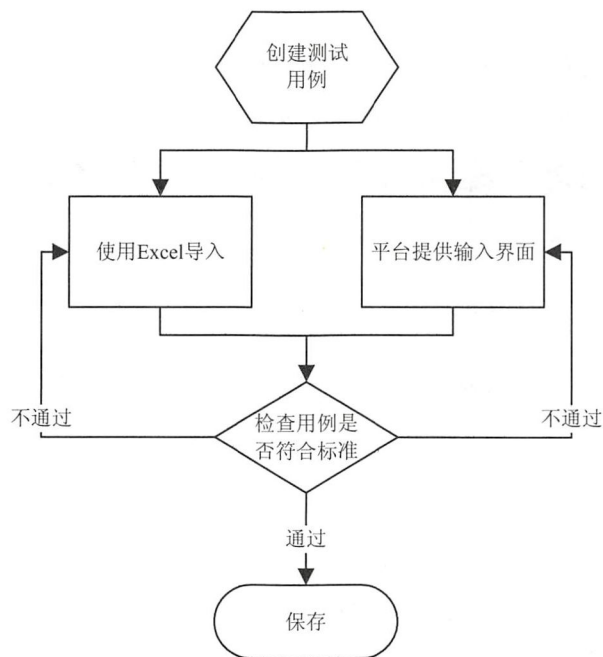


图 8-6 创建测试用例的过程

功能测试人员登录平台，创建测试用例。平台提供两种输入方式，一种是在平台界面中输入，另一种由 Excel 批量导入，保存到数据库中。对于通过平台界面输入的用例，在保存之前，平台会对用例进行完整性检查，通过后方可保存。

创建测试用例需要遵守如下规则：

- 用例中体现的属性有用例名称、用例描述、系统子系统、产品大类子类、功能点

编号、正例反例和备注。

- 业务流是测试用例的骨架，由相互独立的子模块组成。在测试用例里，业务流的顺序将体现功能点编号、实际结果和预期结果等信息。
- 回归测试人员在平台进行测试用例输入时，要注意测试用例分手工测试用例和自动化测试用例，需要在用例中添加检查点。
- 平台对用例进行完整性检查，而用例的合理性和有效性是需要回归测试人员进行检查的。

8.3.2 创建测试数据

由于 Excel 本身具有处理数据的优势，所以本项目使用 Excel 保存测试数据，这有利于数据的维护。新建 Excel 文件，内容如图 8-7 所示。

模块名称	字段名	数据属性	测试数据		
	数据字段名		1	2	3
公共_登录	用户名	数据输入	U001		
	密码	数据输入	111111		
任务中心_流程监控	申请编号	数据输入	15102733100001	1510285113030100001	15102733100001
	客户名称	数据输入	顾慧秀	万曼妮	李艳
	证件号码	数据输入	120000197005233321	120000197303118444	
	当前环节	数据输入	1	1	
任务中心_待分配任务列表	申请编号	数据回写	1510215113030100001	1510215113030100001	1510215113030100001
	客户名称	数据输入	性能测试1	性能测试1	性能测试1
	证件号码	数据输入	110105200510034278	110105200510034278	110105200510034278
	当前环节	数据关联	1510215113030100001		
公共_退出	用户名	数据输入			
	密码	数据输入			

图 8-7 Excel 维护测试数据案例

各列说明如下。

- (1) 模块名称：用于定义模块名称。
- (2) 字段名：定义需要输入数据的名称，脚本中将根据定义的字段名获取测试数据。
- (3) 数据属性分为以下 3 种类型。
 - 数据输入：输入基础数据，如用户名、密码等。
 - 数据回写：这类数据是在脚本运行时产生的，如订单号、流水号等。
 - 数据关联：这类数据是获取上一个业务组件运行结果的数据，如在运行过程中需要用到上一个业务组件生成的订单号。
- (4) 测试数据：1、2、3 列为实际的测试数据。

然后开始配置业务流，业务流由各个模块和数据组成。新建一个 Excel 的 Sheet 页，业

务流程配置如图 8-8 所示，其中定义了任务分配和流程监控两个业务流。

业务流名称	业务流循环次数	模块名称	数据列	是否运行
任务分配	2	公共 登录	1,3	Enable
		任务中心 流程监控	1	Enable
		任务中心 待分配任务列表	1	Enable
		公共 退出	1	Enable
流程监控	3	公共 登录	1	Enable
		任务中心 流程监控	1-3	Enable
		公共 退出	1	Enable

图 8-8 测试业务配置流程

其中各列项含义如下。

- 业务流名称：定义业务流程名。
- 业务流循环次数：定义业务流程运行的次数。
- 模块名称：定义业务流由哪些模块组成。
- 数据列：定义每一次要运行的数据列。
- 是否运行：Enable 表示运行该模块；False 表示不运行该模块。

8.3.3 创建测试项目

具体步骤如下。

(1) 打开 Eclipse，选择“File”→“New”→“Project”菜单项，选择“Java Project”选项，新建项目名称为 ATF，添加项目依赖的 Jar 包，如图 8-9 所示。这些 jar 包的功能如下。

- log4j：日志处理。
- poi：项目中使用的 Excel，用来处理 Excel 的操作。
- reportng：代替 TestNG，并扩展 TestNG 的报表功能。
- sqljdbc4：数据库驱动。
- webdriver：存放 Selenium 2 的 jar 包。
- saxon：解析 XML 文件。
- jxl：配合 poi 解析 Excel 中的数据。

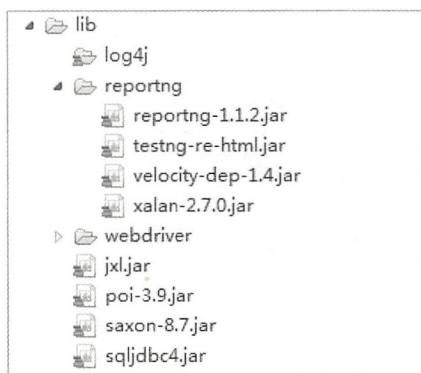


图 8-9 项目依赖的 Jar 包

(2) 由于项目使用 Ant 构建，因此需要添加 build.xml 文件，用于定义项目编译、打包和运行。build.xml 文件内容如下：

```
<?xml version="1.0"?>
<project default="run">
  <property name="sourcepath" value="${basedir}/src" />
  <taskdef resource="testngtasks"
    classpath="lib/testng-re-html.jar"
  />
  <path id="runpath">
    <pathelement location="bin" />
    <path refid="cpath" />
  </path>

  <target name="compilefirst">
    <delete dir="bin" />
    <mkdir dir="bin" />
    <javac includeantruntime="on"
      destdir="bin"
      classpathref="cpath"
      srcdir="src" debug="true"
      encoding="UTF-8">
    </javac>
    <java classpathref="runpath"
      classname="com.test.autotest.util.FileSystem">
      <jvmarg value="-Dfile.encoding=UTF-8" />
    </java>
  </target>
</project>
```

```

</target>

<target name="compile" depends="compilefirst">
    <delete dir="bin" />
    <mkdir dir="bin" />
    <javac includeantruntime="on"
        destdir="bin"
        debuglevel="lines,vars,source"
        classpathref="cpath"
        srcdir="src"
        debug="true"
        encoding="UTF-8"
    >
    </javac>
    <copy file="src/log4j.properties"
        tofile="bin/log4j.properties" />
</target>

<target name="run" depends="compile">
    <delete dir="test-output" />
    <delete dir="test-output/report" />

```

(3) 在 ATF 项目中新建 res 文件夹，用来存放驱动文件。由于框架需要运行在不同的浏览器中，因此需要添加 IE 和 Chrome 的驱动，Selenium 默认包含了 Firefox 驱动，因此不需要单独添加，添加完成后如图 8-10 所示。

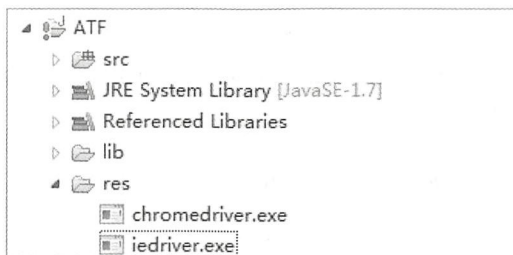


图 8-10 存放驱动文件

(4) 在 ATF 项目的根目录下，新建 Prop.properties 文件，配置运行时的参数，代码片段如下所示：

```
# Browser 类型
```

```
# 1 => FireFox
# 2 => Chrome
# 3 => IE
BrowserCoreType = Chrome
# Drivers 路径
DriverPath = res/chromedriver.exe
# 默认等待时间
Timeout = 20000
#数据库配置
insertSQL = NO
dbUrl = jdbc:sqlserver://127.0.0.1:1433;DatabaseName=Test
dbUser = admin
dbPwd = 1234
```

其中各项参数说明如下：

- BrowserCoreType: 配置要运行的浏览器，可以是 Chrome、IE 或 Firefox。
- DriverPath: 配置要运行的 Driver 路径。
- Timeout: 默认等待时间单位（ms）。
- insertSQL: 是否插入数据库。
- dbURL: 数据库 URL。
- dbUser: 数据库用户名。
- dbPwd: 数据库密码。

(5) 新建 test-xml 文件夹，在该文件夹下新建 atf.xml 文件，用于存放 TestNG 的配置信息，配置信息如下：

```
<suite reruntimes="1" name="demo" thread-count="10" parallel="tests" >
  <test name="demo.xls" >
    <classes>
      <class name="com.test.autotest.run.TestNgRun1" />
    </classes>
  </test>
</suite>
```

(6) 在 ATF 根目录下新建 Action.properties 文件，用于 Excel 中的测试组件和测试脚本相关联，在脚本开发过程中详细介绍该文件的内容。新建 testcase 文件夹用于存放 Excel 数据文件，最终框架结构如图 8-11 所示。

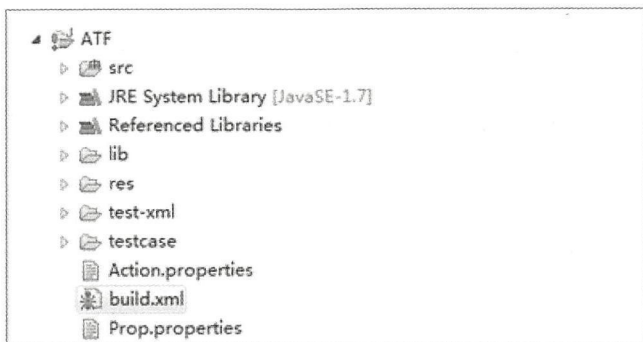


图 8-11 框架结构

8.3.4 开发框架运行类

具体步骤如下。

(1) 新建一个 Java 类读取项目中的配置文件，由于配置文件采用 Key = Value 的方式，例如项目 ATF 中的 Prop.properties 中 “BrowserCoreType = Chrome”，等号左边的 “BrowserCoreType” 为 Key，等号右边的 “Chrome” 为 Key 对应值。该类主要通过以下两个方法来获取配置文件信息。

- `getProperty(String key)`: 通过传入的 Key 值，获取配置文件中 Key 对应的值，实现的代码如下：

```
public String getProperty(String property) {
    //把配置文件转换成 Properties 对象
    Properties prop = getProperties();
    String newProperty="";
    try {
        newProperty = new String(property.getBytes("UTF-8"), "ISO8859_1");
    } catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    //返回 Key 对应的值
    return prop.getProperty(newProperty);
}
```

- `getProperties()`: 把配置文件转换成一个 properties 对象，实现的代码如下：

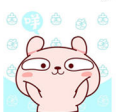
```
public Properties getProperties() {
    Properties prop = new Properties();
    try {
        FileInputStream file = new FileInputStream(fileName);
        prop.load(file);
        file.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return prop;
}
```

(2) 编写框架运行类，此方法将运行一个业务流程，并执行业务流程中包含的组件脚本，该方法传入的参数包含如下。

- FlowInfo：对应 Excel 中的业务流名称。
- RunList：业务流程的运行次数。

传入业务流程和运行次数后，可以运行测试组件，该方法主要代码片段如下：

```
public void doTest(String FlowInfo, int runlist) {
    doBeforeClass();
    long startTime = System.currentTimeMillis();
    SimpleDateFormat createTime = new SimpleDateFormat
("yyyy-MM-dd HH:mm:ss");
    FileUtil fileUtil = new FileUtil();
    String[] FlowStr = FlowInfo.substring(FlowInfo.indexOf("[")
+1,FlowInfo.indexOf("]")).split("-");
    //获取该业务流的起始行
    int left=Integer.parseInt(FlowStr[0]);
    //获取该业务流的结束行
    int right=Integer.parseInt(FlowStr[1]);
    //获取 Excel 路径
    String excelpath=this.NewExcel;
    //定义日志输出路径
    String logfile = "log/"+getClass().getName();
    //删除原来的日志
    fileUtil.deleteFile(logfile);
    try {
        //循环组件
        for(int row = left; row <= right; row++){
            //判断组件是否为 Enable，若是则运行
            String actEnable = excelUtil.ReadExcel
```




```

(excelpath, "Main", 6, row);
    if(actEnable.equalsIgnoreCase("Enable")){
        //获取业务流的测试数据, otestmain 对象存放 Excel 中的测试数据
        int DataList = otestmain.getCurrentDataList
    (excelpath, runlist, row);
        dataList = DataList;
        //获取业务组件名称
        String actionName = excelUtil.ReadExcel
    (excelpath, "Main", 3, row);
        //判断业务流是否运行
        boolean isActRun = otestmain.isActionRun
    (excelpath, actionName, DataList);
        if(!isActRun){continue;}
        driver.SetActionName(excelpath, actionName, dataList);
        String act_name = ActionProp.getProperty(actionName);
        //根据组件名用反射机制调用组件
        try {
            Class action = Class.forName(act_name);
            Class[] pType = new Class[]
                { AutoTestUtil.class,
                    String.class,
                    String.class, int.class
                };
            Constructor ctor = action.getConstructor(pType);
            Object obj[] = new Object[]
                {driver, excelpath, actionName, DataList };
            Object actobj = ctor.newInstance(obj);
            //运行组件, 执行页面操作
            Method mthd = actobj.getClass().getMethod("commonMthd");
            //执行 commonMthd 方法
            mthd.invoke(actobj);
        } catch (Exception e)
        {
            e.printStackTrace();
            Assert.fail("use reflect to call action fail!");
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

```



8.3.5 开发公共接口

具体步骤如下。

(1) 由于框架运行在不同的浏览器，因此需要创建一个驱动公共类，实例化不同的浏览器：

```
public WebDriver getWebDriverClient() {  
    //从配置文件获取浏览器类型  
    String browserType = DriverProp.getProperty("BrowserCoreType");  
    if (webdriver == null) {  
        if (browserType.equals("Firefox")) {  
            try  
            {  
                //启动 Firefox  
                webdriver = new FirefoxDriver();  
                return webdriver;  
            } catch (Exception e)  
            {  
                logger.info("启动异常: "+e.getMessage());  
            }  
        }  
        if (browserType.equals("Chrome")) {  
            try {  
                //启动 Chrome  
                System.setProperty("webdriver.chrome.driver",  
                    DriverProp.getProperty("DriverPath"));  
                webdriver=new ChromeDriver();  
                return webdriver;  
            } catch (Exception e)  
            {  
                logger.info("启动异常: "+e.getMessage());  
            }  
        }  
        if (browserType.equals("IE")) {  
            try {  
                //启动 IE  
                System.setProperty("webdriver.ie.driver", DriverProp.getProperty("DriverPath"));  
                DesiredCapabilities dc = DesiredCapabilities.internetExplorer();  
                dc.setCapability(  

```



```

InternetExplorerDriver
.INTRODUCE_FLAKINESS_BY_IGNORING_SECURITY_DOMAINS, true);
//IE 默认启动保护模式，关闭保护模式，
dc.setCapability("ignoreProtectedModeSettings", true);
webdriver = new InternetExplorerDriver(dc);
webdriver.manage().window().maximize();
return webdriver;
} catch (Exception e)
{
    logger.info("启动异常: "+e.getMessage());
}
}
}
return webdriver;
}

```

(2) 由于测试工程师在脚本开发中会用到 WebDriver API，需要对 WebDriver API 的基本方法进行二次封装，便于测试工程师开发脚本。

(3) 脚本中会使用到不同类型的数据库操作，包含 MySQL、Oracle、SqlServer 等数据库操作，需要一组统一的方法封装各种数据库的增加、删除、修改和查询操作，具体实现此处不再描述。

(4) 在脚本开发中需要从 Excel 中获取数据或者回填数据，可以使用下面 3 种方法来处理 Excel 的数据。

- **GetItem(String fileName)**。从 Excel 中获取相应字段的数据，fileName 为字段名称：

```

public String GetItem(String fieldName){
    int row=0;
    try {
        row = hashmap.get(fieldName);
    } catch (Exception e) {
        driver.handleFailure("获取数据字段失败: "+fieldName);
        e.printStackTrace();
    }
    return excelUtil
        .ReadExcel(testCasePath,sheetName,col,row);
}

```

- **SetItem(String fileName,String setData)**。回填数据到字段名为 fileName 的 Excel：




```
public void SetItem(String fieldName, String setData) {
    int row = 0;
    try {
        row = hashMap.get(fieldName);
    } catch (Exception e) {
        driver.handleFailure("回填数据字段为: "+fieldName);
        e.printStackTrace();
    }
    excelUtil.UpdateExcel(testCasePath, testCasePath, sheetName, col, row, setData);
}
```

- `isExist(String FileName)`。判断 Excel 中是否有 `fileName` 这个字段名称：

```
public boolean isExist(String fieldName) {
    return hashMap.containsKey(fieldName);
}
```

(5) 编写测试脚本需要继承的基础类，其中包含该组件的数据。因为业务不同，需重写 `CommonMethod()` 方法，因此在本文实例中，并未具体实现 `CommonMethod()` 定义业务组件的操作。代码如下所示：

```
public class Action {
    String testCasePath;
    String sheetName;
    public oTestData otestdata;
    public MyWebDriver MyDriver;
    public Page page;
    /**
     * 在构造函数中初始化测试数据、驱动程序、公共方法等实例
     * @param driver 驱动程序
     * @param testCasePath 测试数据路径
     * @param sheetName Excel 中测试组件名称
     * @param dataList 测试数据
     * @return
     */

    public Action(AutoTestUtil driver, String testCasePath, String sheetName, int dataList) {
        this.testCasePath = testCasePath;
        this.sheetName = sheetName;
        HashMap<String, Integer> hm = getRowWithTestData
        (testCasePath, sheetName);
    }
}
```





```
//实例化 WebDriver 对象
MyDriver = new MyWebDriver(driver);
page= new Page(MyDriver);
//实例化测试数据
otestdata = new oTestData
(testCasePath,sheetName,dataList,hm);
}

/**
 * 获取数据字段所在行的位置，返回
 * @param testCasePath
 * @param sheetName
 * @return
 */
public HashMap<String, Integer> getRowWithTestData
(String testCasePath,String sheetName){
ExcelUtil excelUtil = new ExcelUtil();
String testData="";
int rows=0;
HashMap<String, Integer> hm= new HashMap<String, Integer>();
do {
    //根据 Excel 定义的业务流程，获取测试数据
    testData= excelUtil.ReadExcel
(testCasePath, sheetName, 0, rows);
    hm.put(testData, rows);
    rows=rows+1;
} while (
!excelUtil.isEmpty(testCasePath, sheetName, 0, rows)
);
    return hm;
}
/**
 *需要组件脚本重写该方法
 * @return
 */

public void CommonMethod()
{

}
}
```





8.3.6 添加日志报告

在项目中会处理大量的日志，可以使用 Apache 的 log4j 日志框架来处理，使用前需要添加 log4j.properties 进行配置，配置代码如下：

```
log4j.rootCategory=INFO, toConsole, toFile
log4j.logger.toConsole = INFO
log4j.appender.toConsole=org.apache.log4j.ConsoleAppender
log4j.appender.toConsole.layout=org.apache.log4j.PatternLayout
log4j.appender.toConsole.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%t]-[%p] %m%n
log4j.logger.toFile = INFO
log4j.appender.toFile=org.apache.log4j.DailyRollingFileAppender
log4j.appender.toFile.file=fileLog
log4j.appender.toFile.append=true
log4j.appender.toFile.encoding=utf-8
log4j.appender.toFile.layout=org.apache.log4j.PatternLayout
log4j.appender.toFile.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%t]-[%p] %m%n
```

8.4 脚本开发

自动化脚本的开发调试，是整个自动化测试流程中不可或缺的一部分。优质而稳定的自动化脚本，可以减少脚本后期的维护成本，并保证每次测试执行过程中的稳定性。

8.4.1 编写测试脚本

具体步骤如下。

(1) 新建一个存放测试脚本的包 com.test.autotest.testcase，创建业务测试脚本 LoginAction，它继承前面开发框架编写的 Action 类，因为 Action 类中包含了测试数据、测试驱动程序和公共方法，可以在 LoginAction 中使用这些公共方法。该测试脚本需要重写 CommonMethod 方法，该方法主要实现业务组件的功能，代码如下：

```
publicclass LoginAction extends Action {

    public LoginAction(AutoTestUtil driverPar, String testCasePath,
        String sheetName, int dataList) {
        super(driverPar, testCasePath, sheetName, dataList);
        // TODO Auto-generated constructor stub
    }
}
```





```
}

@Override
public void CommonMethod()
{
    //需要访问的 Url
    String Url = "http://127.0.0.1:8080/app/";
    //打开浏览器，在浏览器中输入 Url
    MyDriver.get(Url);
    //等待 id 为 j_username 的元素出现
    MyDriver.WaitForElement(By.id("j_username"));
    //在用户名的输入框中输入从 Excel 中获取的用户名
    MyDriver.Type(By.id("j_username"), otestdata.GetItem("用户名"));
    //在密码的输入框中输入从 Excel 中获取的密码
    MyDriver.Type(By.id("j_password"), otestdata.GetItem("密码"));
    //单击提交按钮
    MyDriver.Click(By.xpath("//button[@id='btnSubmit']"));
}
}
```

(2) 配置脚本和 Excel 定义组件的对应关系，通过 Action.properties 属性文件使测试脚本和 Excel 中的组件一一对应，Action.properties 代码片段如下，等号左侧为 Excel 中的业务组件名称，右侧为对应的测试脚本，这样测试脚本和 Excel 中的业务组件名称关联起来：

```
公共_登录 = com.test.autotest.testcase.LoginAction
```

8.4.2 调试运行脚本

具体操作如下。

(1) 使用 ANT 运行项目中所有的测试脚本，在 Eclipse 中用鼠标右键单击 build.xml 文件，选择“Run as”→“Ant Build”选项，即可读取测试管理器数据，编译业务组件代码，生成测试用例文件。

(2) 脚本调试需要注意的以下几点：

- 调试时建议一次只对单个用例进行调试，不使用并发运行。
- 运行过程中每步操作都有控制台输出，可以实时监控执行过程，报错输出也会在控制台打印。
- 修改 Log4J 日志级别，可以查看详细日志。





(3) 执行结果。

- 运行结束后会生成日志文件 fileLog，可以查看运行日志。
- 若运行过程报错，错误截图会以时间命名并保存在 screenshot 文件夹下。
- 项目目录下的 test-output 文件夹中会生成单次测试报告，可以查看。
- 若配置执行结果插入数据库，则可在数据库中查看执行结果及报错信息。

8.4.3 上传脚本

本项目采用 Git 版本控制工具来管理脚本，调试编译通过后把脚本上传到 Git 服务器。在使用 Git 之前，需要安装 Eclipse 的 EGit 插件，安装 EGit 插件的方法可参考 TestNG 的安装，EGit 地址是：<http://download.eclipse.org/egit/updates>。

(1) 安装好 EGit 插件，选择需要上传的项目，单击鼠标右键，在弹出菜单中选择“Team”选项，单击“Share Project”选项，出现如图 8-12 所示 Share Project 界面的，选择 Git 选项，再单击“Next”按钮。

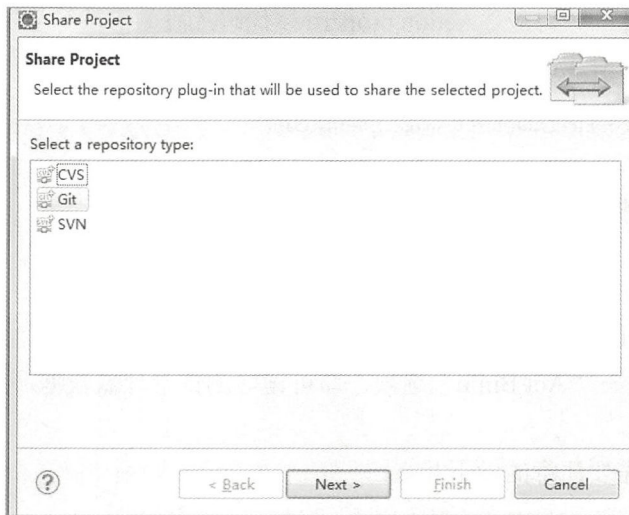


图 8-12 选择代码库 Git

(2) 开始创建一个 Repository，如图 8-13 所示，选择项目，然后单击“Finish”按钮，创建 Repository 成功。

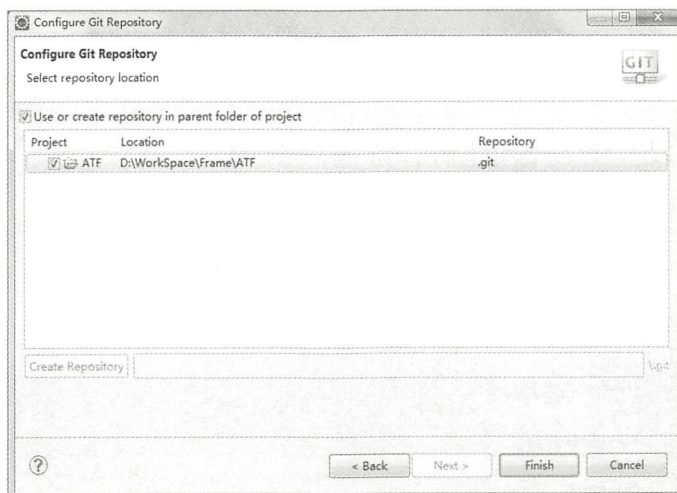


图 8-13 创建 Repository

(3) 选择项目，单击右键，选择“Team”→“Commit”选项，出现如图 8-14 所示的提交代码窗口，选择要提交的文件，在“Commit Message”输入框中输入本次提交代码的原因及修改的内容，单击“Commit and Push”按钮开始提交代码到 Git 服务器。

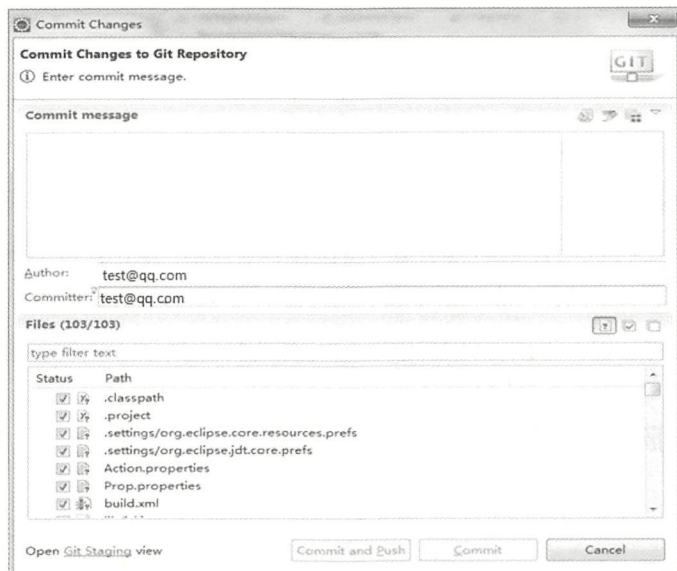


图 8-14 选择提交代码窗口



(4) 单击“Commit and Push”按钮后，在如图 8-15 所示的界面上，输入项目在 Git 上的 URL、用户名和密码，单击“Finish”按钮提交代码，完成代码的提交。

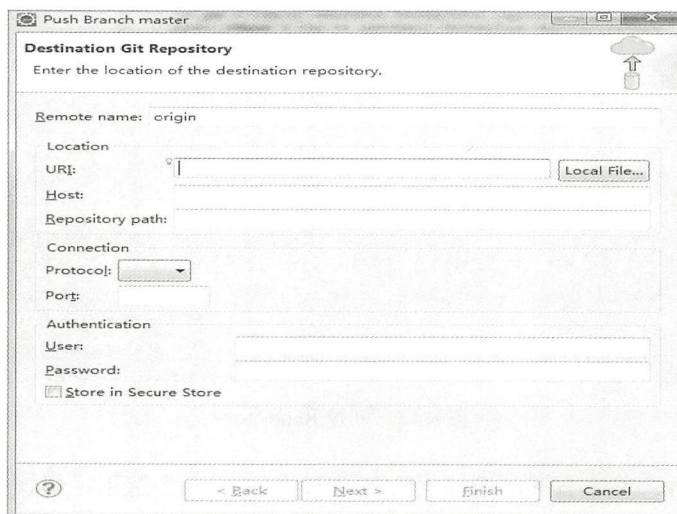


图 8-15 提交代码

8.5 持续集成

开发工程师和自动化测试工程师完成本地计算机代码构建后把代码提交到代码库 SVN 或 Git 中，持续集成平台获取测试脚本，完成脚本的编译，执行自动化单元测试，通过 Sonar 完成代码静态扫描，将代码打包后部署到不同的测试环境，根据定时任务进行自动化集成测试，执行完自动化测试后，生成测试报告，通过邮件发送测试结果给测试人员和开发人员，这是整个持续集成的过程。下面介绍使用 Jenkins 作为持续集成工具实施自动化集成测试。

8.5.1 Jenkins 服务器搭建

持续集成的环境搭建采用 Jenkins 作为持续集成的服务器，下面讲述搭建配置 Jenkins 服务器的步骤。

(1) 下载 Tomcat 安装包，解压安装包放置到 C 盘，如图 8-16 所示。



- (2) 添加环境变量: CATALINA_HOME=C:\tomcat。
- (3) 下载 Jenkins 的 WAR 包, 拷贝到 C:\tomcat\webapps 目录下。
- (4) 双击 C:\tomcat\bin\startup.bat 文件, 如图 8-17 所示, 启动 Tomcat 服务器。



图 8-16 Tomcat 目录结构

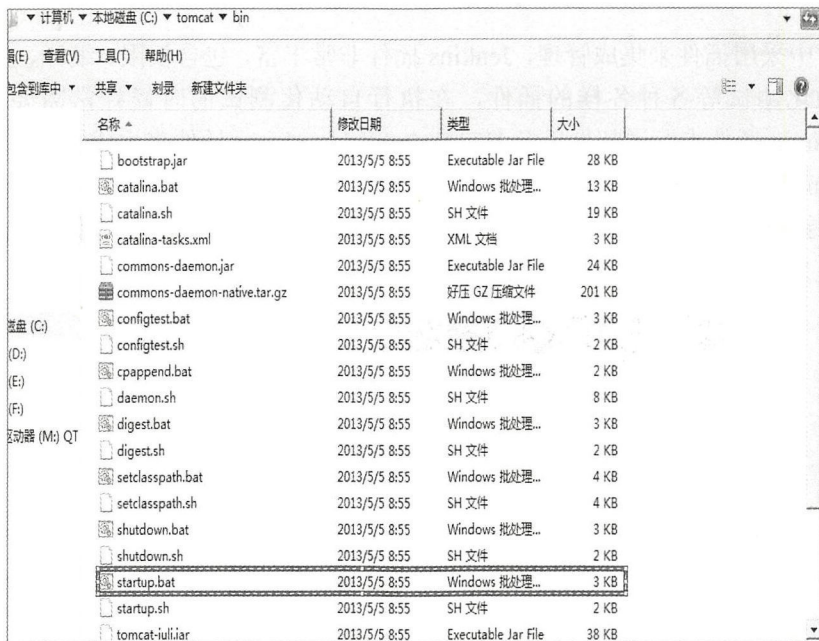


图 8-17 启动 Tomcat

- (5) 访问 <http://127.0.0.1:8080>, 进入 Jenkins 主界面, 如图 8-18 所示。



质量全面管控：从项目管理到容灾测试



图 8-18 Jenkins 主界面

8.5.2 Jenkins 相关插件

Jenkins 中采用插件来集成管理，Jenkins 插件非常丰富，包含权限、编译、打包、报表、邮件和自动化测试等各种各样的插件，在执行自动化测试的时候经常需要安装 Email Extension Plugin 插件来发送邮件以及 HTML Publisher plugin 插件来生成 HTML 报告。下面开始介绍 Jenkins 中的插件管理。

(1) 登录 Jenkins 服务器首页，选择左侧的“系统管理”→“管理插件”选项，然后在界面中单击“可选插件”选项卡，如图 8-19 所示。



图 8-19 Jenkins 插件



(2) 在“Filter”文本框中输入要查找的插件，在返回结果列表中选择需要安装的插件后，单击“直接安装”按钮，Jenkins 会自动安装插件。

(3) 安装成功后可以在“已安装”选项卡中查看该插件，可以设置是否启用，也可以在该界面卸载不需要的插件如图 8-20 所示。



图 8-20 Jenkins 插件界面

8.5.3 部署测试执行机

为了提高测试效率，缩短测试时间，可以采用多台机器执行，这些机器称为测试执行机，主要负责执行测试任务。需要对测试执行机进行如下配置，才能开始执行测试任务。

(1) 由于项目采用 Ant 构建和运行，因此需要先安装 Ant，Ant 安装步骤如下所示。

- 下载 Ant 安装包后，解压到 C:\apache-ant-1.9.6。
- 添加系统环境变量如下所示：
 - ANT_HOME=C:\apache-ant-1.9.6
 - Path= C:\apache-ant-1.9.6\bin
- 在 Windows 命令行输入 ant -v，可以查看 Ant 是否安装成功，成功信息如图 8-21 所示。

```
C:\Users\Administrator>ant -v
Apache Ant(TM) version 1.9.6 compiled on June 29 2015
Trying the default build file: build.xml
Buildfile: build.xml does not exist!
Build failed
```

图 8-21 Ant 安装成功



质量全面管控：从项目管理到容灾测试

(2) 由于项目需要在不同的浏览器中运行，因此在测试执行机中安装 Firefox、IE 和 Chrome 来满足不同浏览器的兼容测试。

8.5.4 分布式测试

为了提高测试效率可以采用分布式测试，通过 Jenkins 管理测试执行机，使测试脚本可以运行在不同的执行机上，Jenkins 的管理节点模块可以添加执行机，并且收集执行机运行结果，实现自动化分布式测试。下面将介绍如何通过 Jenkins 来管理测试执行机。

(1) 登录 Jenkins 服务器首页。如图 8-22 所示，选择左上角“系统管理”菜单，单击“管理节点”链接。

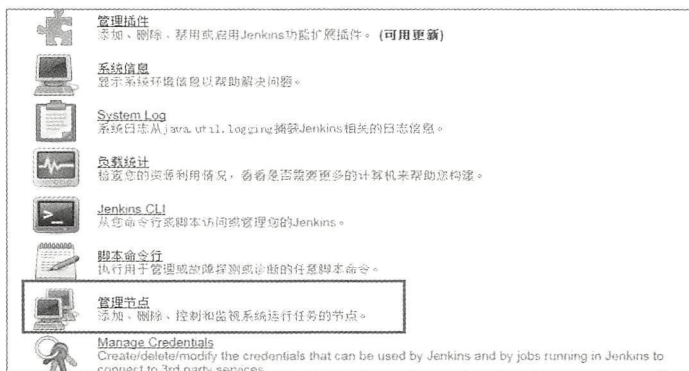


图 8-22 Jenkins 管理节点

(2) 进入管理节点界面，如图 8-23 所示，选择“新建节点”选项。若将执行机与节点关联，则可以在执行机上执行与其关联的节点任务。

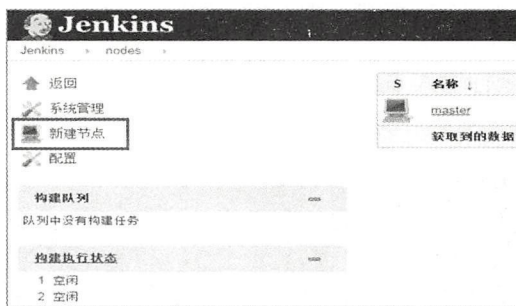


图 8-23 新建节点入口

(3) 新建节点界面如图 8-24 所示，在“节点名称”输入框中输入节点的名称，该名称用来标识该执行机。然后选择“Permanent Agent”选项，单击“OK”按钮进入下一个页面。

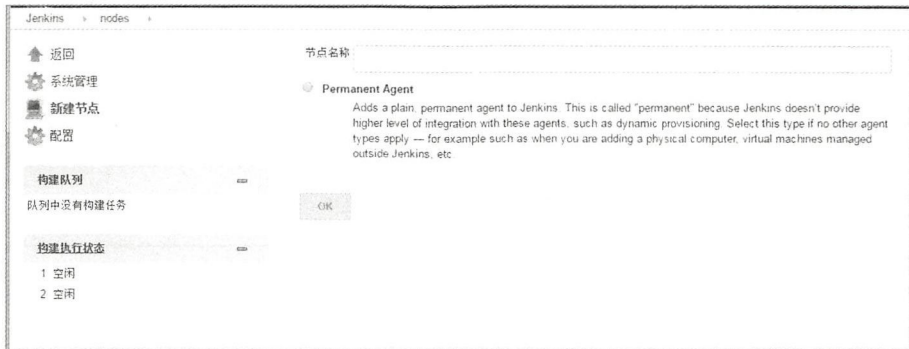


图 8-24 新建节点界面

(4) 远程工作目录是执行机的工作目录。如果执行机是 Windows，则“启动方法”选项应选择“Launch slave agents via Java Web Start”。最后单击“Save”按钮，保存成功之后，执行机在 Jenkins 上的配置就完成了，如图 8-25 所示。



图 8-25 配置节点界面

(5) 在执行机管理页面，可以在执行机列表中查看到已经添加的执行机，如图 8-26 所示，图中的执行机（Test）状态为：未启动。

质量全面管控：从项目管理到容灾测试

S	名称	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	11.48 GB	6.00 GB	11.48 GB	0ms
	Test		N/A	N/A	N/A	N/A	Time out for last 1 try
	获取到的数据	2 分 49 秒	2 分 49 秒	2 分 49 秒	2 分 49 秒	2 分 49 秒	2 分 49 秒

图 8-26 查看配置的节点

(6) 单击鼠标右键选择要启动的执行机，进入节点启动界面，如图 8-27 所示，单击“Launch agent”按钮，既可以启动执行机。



图 8-27 启动节点

(7) 在启动执行机过程中，会弹出如图 8-28 所示的安全警告提示框，可以忽略此安全警告，勾选“我接收风险并希望运行此应用程序”选项，单击“运行”按钮，继续运行。

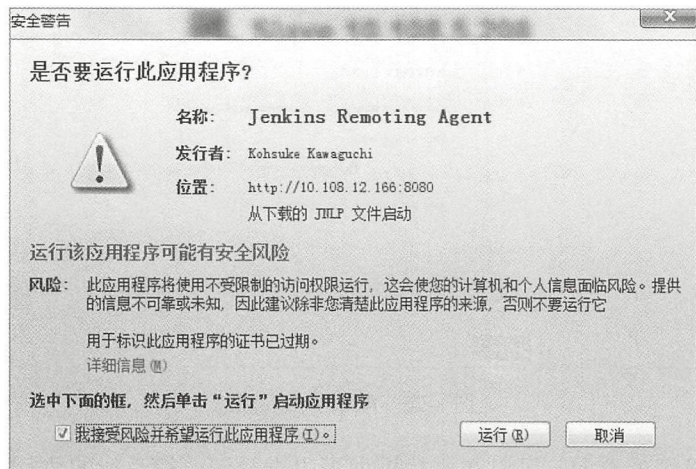


图 8-28 警告提示

(8) 节点连接成功, 如图 8-29 所示。注意在执行机运行过程中不能关闭此对话框, 如果需要开机启动, 可以选择菜单“File”里的“Install as Services”选项作为 Windows 开机服务。



图 8-29 成功连接节点

8.5.5 配置测试任务

在 Jenkins 中采用任务的方式管理项目, 这个任务主要配置该项目的基本信息、源代码获取方式、构建方式和构建后的操作, 配置成功后, Jenkins 就可以根据配置信息来执行该任务, 下面介绍 Jenkins 中的任务配置。

1. 新建任务

(1) 登录 Jenkins 服务器首页, 将鼠标单击左侧“新建”选项, 新建一个任务, 如图 8-30 所示。

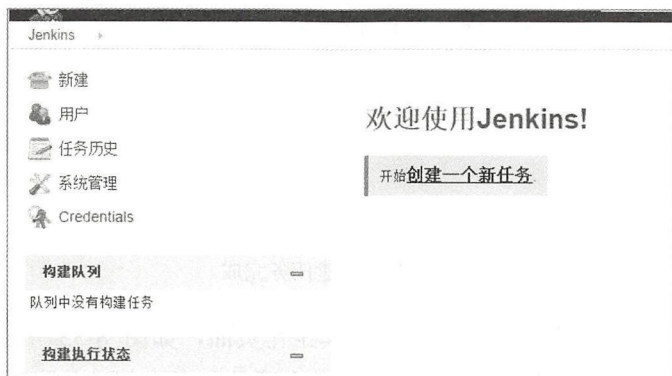


图 8-30 新建任务入口

(2) 进入新建任务界面, 如图 8-31 所示, 输入项目名称: Test_demo, 选择“构建一

质量全面管控：从项目管理到容灾测试

个自由风格的软件项目”选项，然后单击“OK”按钮，就完成了新建任务。

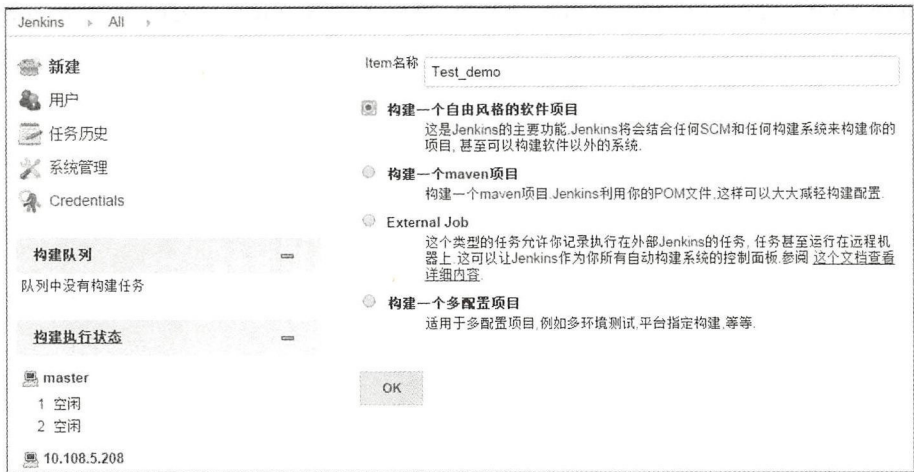


图 8-31 新建任务界面

2. 基本配置

(1) 通过上节完成新建任务之后，返回 Jenkins 主界面，在主界面可以看到刚刚新建的任务如图 8-32 所示。

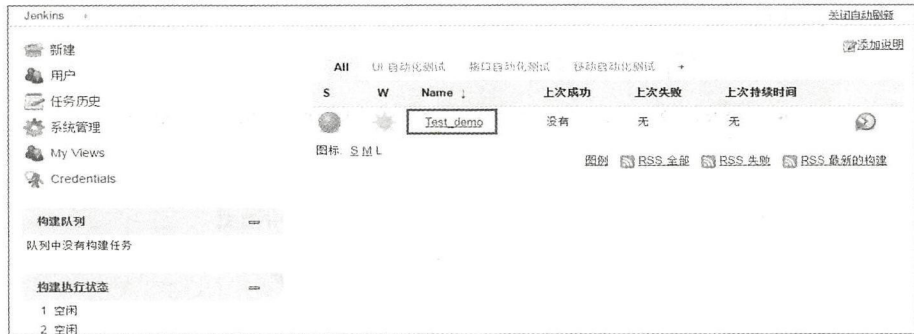


图 8-32 新建任务完成

(2) 单击“Test_demo”这个任务，进入其工作页面，如图 8-33 所示。单击左侧“配置”选项，就可以进入任务基本信息配置页面。主要的任务有基本信息配置、获取源管理、构建触发器、构建和构建后的操作。

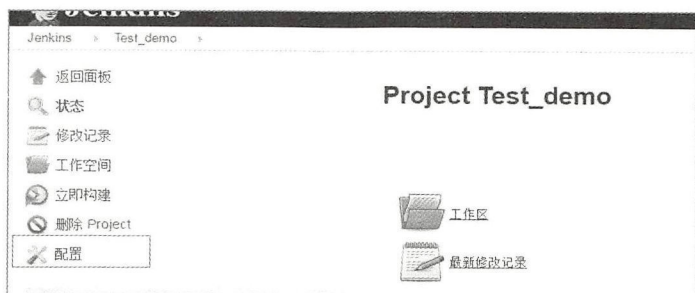


图 8-33 任务配置界面

(3) 基本信息配置主要包含项目名称、描述和显示名称。通过项目名称可以区分不同的项目；显示名称可以为该任务分配执行机的标签。如图 8-34 所示为任务基本配置页面。

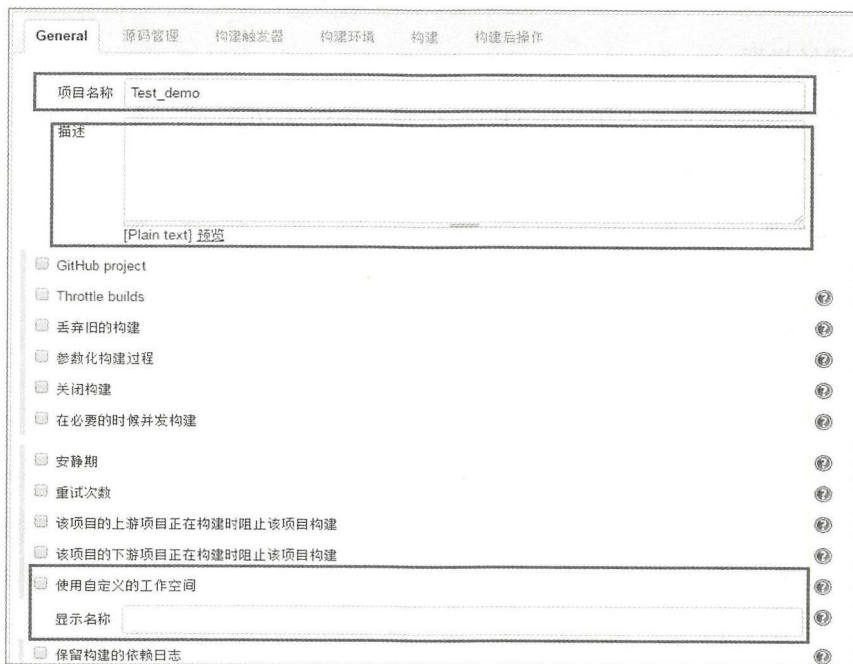


图 8-34 任务基本配置页面

3. 源代码管理

Jenkins 源代码通过 CVS、SVN、Git 等方式进行管理，每次构建时将会从这些服务器上下载源代码。如图 8-35 所示为源码管理页面，图中源代码管理工具选择 Git。

质量全面管控：从项目管理到容灾测试

源码管理

☐ None
☐ CVS
☐ CVS Projectset
☒ Git

Repositories

Repository URL:

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

图 8-35 源码管理页面

4. 构建触发器

构建触发器主要实现无人值守下的自动构建，如图 8-36 所示，通过配置构建日程表，每天 17:00 都会自动构建运行。

构建触发器

☐ 触发远程构建 (例如, 使用脚本)

☐ Build after other projects are built

☒ Build periodically

日程表:

☐ No schedules so will never run

☐ Build when a change is pushed to GitHub

☐ Poll SCM

图 8-36 运行策略

5. 构建

(1) Jenkins 构建方式可以使用 Maven、批处理、Shell、ANT 等多种方式进行项目构建，本项目采用 Windows Batch Command 来构建，在“增加构建步骤”下拉框中选择 Execute Windows batch command，如图 8-37 所示。



图 8-37 脚本构建方式

(2) 弹出 Execute Windows batch command 的输入框，如图 8-38 所示。在输入框中可以输入构建的命令，这些命令与 Windows 系统的批处理命令类似。

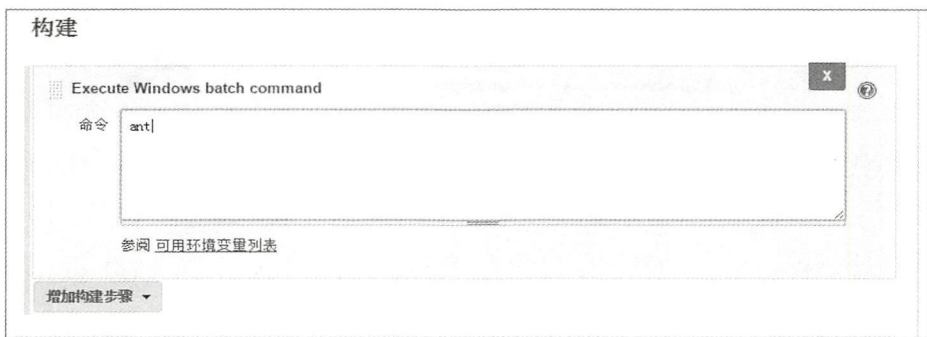


图 8-38 输入 ant 命令

6. 构建后操作

(1) 构建完成之后，如果要发送邮件和生成 HTML 报告，需要单击“增加构建后操作步骤”选项，添加“Public HTML reports”、“E-Mail Notification”和“Editable E-Mail Notification”3 个配置项，如图 8-39 所示。

(2) 通过 Public HTML reports 配置生成的 HTML 测试报告，需要配置 index.html 存放目录，配置界面如图 8-40 所示。

(3) 如果构建失败发送错误信息到联系人的邮箱，输入联系人邮箱 Test@qq.com，配置界面如图 8-41 所示。

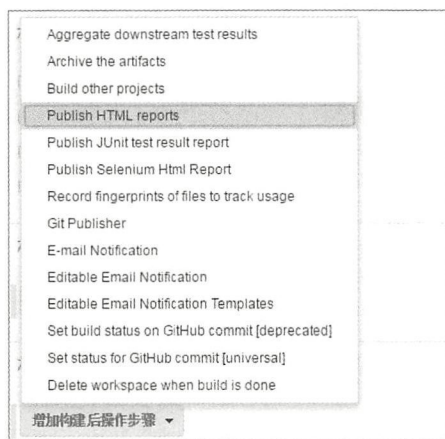


图 8-39 构建完成输出 HTML 报告

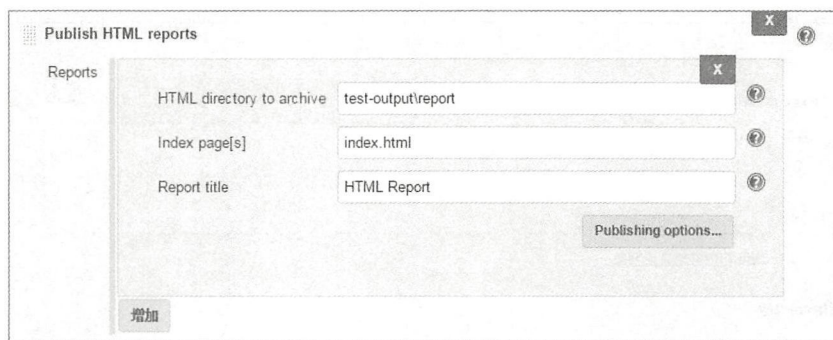


图 8-40 配置 HTML 报告存放目录

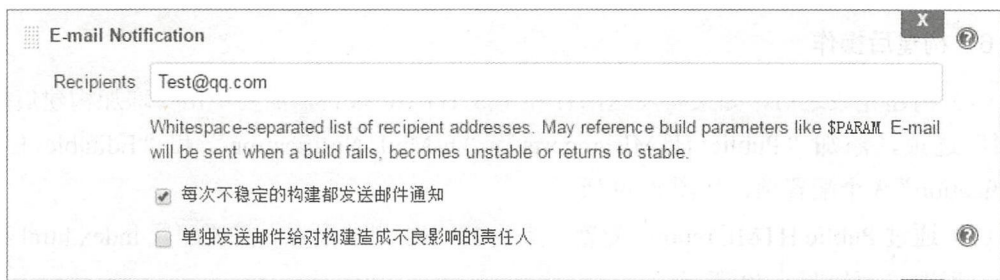


图 8-41 构建失败邮件通知

(4)如果构建成功,需要发送生成好的 HTML 报告到联系人的邮箱,配置界面如图 8-42 所示。

Editable Email Notification

☐ Disable Extended Email Publisher
Allows the user to disable the publisher, while maintaining the settings

Project Recipient List
\$DEFAULT_RECIPIENTS
Comma-separated list of email address that should receive notifications for this project.

Project Reply-To List
\$DEFAULT_REPLYTO
Comma-separated list of email address that should be in the Reply-To header for this project.

Content Type
Default Content Type

Default Subject
\$DEFAULT_SUBJECT

Default Content
\$DEFAULT_CONTENT

Attachments

Can use wildcards like 'module/dist/*'.zip' See the @includes of Ant fileset for the exact format. The base directory is the workspace.

Attach Build Log
Do Not Attach Build Log


Content Token Reference

图 8-42 HTML 报告发送邮箱配置

下面对几个主要的参数做简要说明。

- **Project Recipient List:** 配置收件人邮箱;
- **Content Type:** 指定构建后发送邮件内容的类型, 有 Text 和 HTML 两种;
- **Default Subject:** 自定义邮件通知的默认主题名称。该插件可以在邮件的主题字段中替换一些令牌, 这样就可以从构建中包含指定的输出信息;
- **Default Content:** 自定义邮件通知的默认内容主体。该插件可以在邮件的内容主体中替换一些令牌, 这样就可以从构建中包含指定的输出信息。

7. 执行任务

(1) 进入 Jenkins 首页, 主界面上显示了刚刚已经配置好的任务, 单击  按钮, 将执行构建任务, 如图 8-43 所示。

All 创建自动化测试 创建配置 创建测试 管理自动化 自动化测试 保存历史						
S	W	Name	上次成功	上次失败	上次持续时间	
		BOE2Web	1月24 days - #235	1月24 days - #235	85秒	
		demo_01	6 days 21小时 - #21	20 days - #51	1分14秒	
		BOE2Web-Sonar	2月1 day - #37	7分0秒 - #257	2分0秒	
		BOE2Web-Sonar	2月1 day - #100	16小时 - #222	5分26秒	
		Jsonnet-Demo	12月 - #233	1月0 days - #155	92秒	
		Sonar-Demo	2月1 day - #462	11分 - #532	1分27秒	
		Test_demo	44分 - #235	2月29 days - #51	18秒	
		Test_demo_2	20分 - #235	1月0 days - #155	29秒	

图 8-43 选择构建任务

(2) 还有另外一种方式可以构建任务：进入该项目的主面板，如图 8-44 所示，鼠标单击“立即构建”选项也可以构建任务。



图 8-44 项目主面板

8.5.6 查看运行结果

Jenkins 提供一个控制台来查看运行日志，控制台详细记录了 Job 运行过程中的信息。通过该控制台打印的提示信息，可以查看任务是否运行成功并分析任务失败的原因，也可以通过自动化测试报告查看用例运行情况，如总用例数、失败用例数和成功用例数。下面将介绍使用 Jenkins 如何查看测试运行结果。

(1) 通过 Jenkins 服务器的 Console Output 面板可以查看到已经构建完成的项目指定日期的日志，如图 8-45 所示。

(2) 如图 8-46 所示，单击“自动化测试报告”链接，可查看本次构建生成的 HTML 报告，只有配置了 HTML 报告的输出，才会有该报告生成。

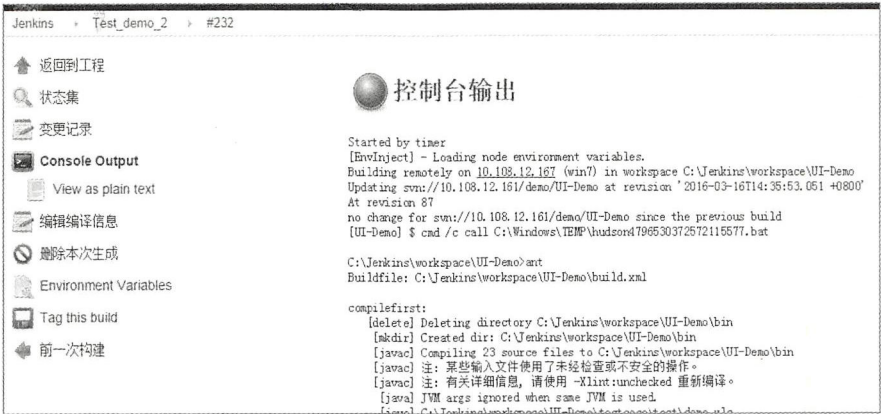


图 8-45 控制台日志输出

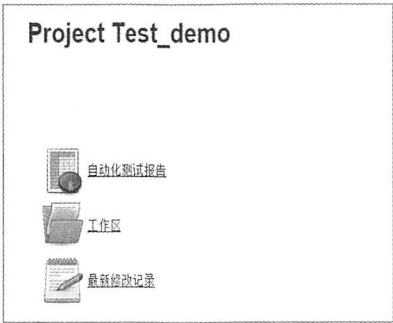


图 8-46 运行结果

(3) 查看 HTML 报告，主要包含的内容有总用例数 (All)、失败用例数 (Fail) 和成功用例数 (Pass)，如图 8-47 所示，该报告会自动发送到 8.5.5 节中已经配置好的联系人邮箱。

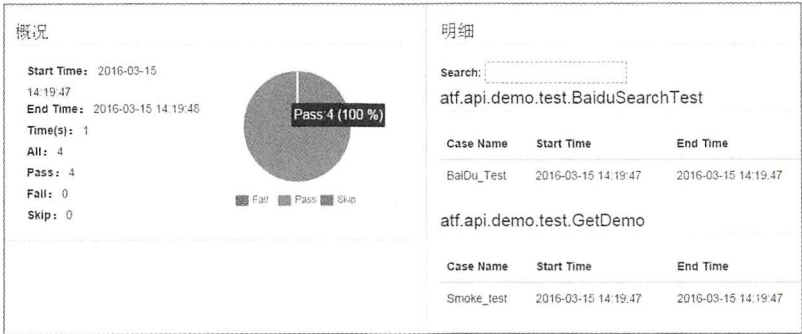


图 8-47 测试结果报告

8.6 要点回顾

通过本章的学习，了解了采用业务流程分析方法进行自动化测试框架的搭建过程。学习了如何设计测试框架，包含测试用例的维护、框架的分层和框架的工作流程。了解了如何开发框架，包含公共组件开发、框架运行类、生成测试报告以及测试数据的维护等。了解了完成测试框架开发后需要根据业务编写测试脚本，并上传脚本到服务器。学习了采用 Jenkins 做持续集成，并利用分布式测试提供测试效率。

测试框架的中心思想是测试数据和脚本相分离，即使测试数据改变也不需要修改测试脚本，通过分层的思想设计测试脚本，降低测试脚本的依赖性。自动化测试的目标是帮助测试人员进行回归测试，通过自动化测试节约人力，缩短测试时间，在有限的时间中覆盖更多的测试用例，保证系统的平稳上线。

第 9 章

性能测试

一辆酷炫的跑车在基础的功能测试结束后，需要对其进行性能测试。

- 车子最快的速度是多少？
- 车子最省油时的速度是多少？
- 轮胎最多能使用多久？
- 什么情况下轮胎会抱死，如何优化此问题？
- 使用油表盘、速度表和其他行车电脑仪来观测车子的状态

那么，如何来测试和定义这些指标呢？本章将讲解性能测试的作用和方法。

- 性能测试的术语。
- 性能需求分析。
- 制定测试策略。
- Load Runner 和 JMeter 的使用。
- 性能测试框架（JMeter、InfluxDB、Grafana）。
- 性能测试实战。
- 常见的性能测试调优方案。

9.1 性能测试基础

性能测试（Performance Testing）是模拟生产环境中的关键业务压力和多种场景，在多种正常、峰值和异常负载条件下，对系统的各项性能指标进行测试，判定其是否满足系统

性能的需求。性能测试属于性能工程（Performance Engineering）的一个子集，性能工程还包括对性能测试发现的性能问题进行定位调优。

性能测试概括为 3 个方面：应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。通常情况下，通过 3 个方面有效、合理的结合，可以实现对系统性能的全面分析和瓶颈的预测。

9.1.1 性能术语

本节旨在解释性能测试中经常出现的一些名词，有助于读者的理解。

1. 用户数（Users）

有时会看到下面这样的描述：一个系统注册用户达到 6000 万人，其中每小时的活跃用户大概在 60 万人左右。这段描述介绍了两个信息，第一个信息：6000 万人指的是注册用户，第二个信息：60 万人指的是真实在线用户。

- 注册用户

注册用户是存在于系统数据库表中的基础数据。这部分用户是指系统所拥有的所有用户群体。这些用户是不会全部对系统造成压力的，唯一的压力就是这些用户占用了系统的存储，影响了数据库的容量。

- 在线用户

在线用户是真正产生压力的用户，这些用户是压力的根源，也就是系统要能够支持这么多人同时在线业务。

- 并发用户

在线用户是真实的用户，但不是所有的在线用户都会在系统上做操作，可能有些用户在浏览网页、有些用户在做业务、有些用户只是开着浏览器。这时在线用户中对系统产生压力的用户只是一部分，而这部分用户就是在线用户中的有效并发用户。

- 虚拟用户

虚拟用户即 Virtual User，简称 Vuser，是性能测试工具产生的用户（JMeter 和 Load Runner 都可以产生虚拟用户），用来模拟真实用户进行的一系列业务逻辑操作。

2. 事务（Transaction）

事务指的是业务逻辑上的事务。例如注册用户是一个事务，指用户提交注册发生时到

注册成功返回页面时的事务，它包括多个页面或者 JS 等组件的交互。一般的响应时间都是指事务从开始到结束的时间。

3. 响应时间 (Response Time)

响应时间是指从客户端发出一个请求开始计时，到客户端接收到服务端的响应结果为结束所经历的时间。响应时间由请求发送时间、网络传输时间和服务器响应时间 3 部分组成。

在性能测试结果分析中，分为事务最小响应时间、事务平均响应时间、事务最大响应时间和 90%事务响应时间。一般使用的标准时间为 90%响应时间，即测试过程中 90%的业务用了多长时间。

例如，一个小时内，查询业务进行了 1000 次，其中 900 次查询业务的响应时间是 1 秒，100 次查询业务的响应时间是在 1 秒上下浮动，那么 90%响应时间为 1 秒。

4. 每秒事务数 (TPS、QPS、PV)

在性能测试中，经常会用到 TPS、QPS、PV。其中，TPS、QPS 在性能测试中有时可以认为是一样的，性能测试工具（比如 Load Runner）是以计算 TPS 作为性能指标的，Load Runner 打开一个页面或者发送一个 HTTP、Sockets 请求，也就是产生一个事务。这个事务可以是查询业务，也就是 QPS。一个页面可能包含多个图片，也就是有多个 PV。

● QPS

QPS 即 Queries Per Second，意思是“每秒查询率”，是一台服务器每秒能够响应的查询次数，是对一个查询服务器在规定时间内所处理流量多少的衡量标准。

● TPS

TPS 即 Transaction Per Second，意思是每秒事务数，它是性能测试结果的衡量单位。一个事务是指一个客户机向服务器发送请求后服务器做出反应的过程。客户机在发送请求时开始计时，收到服务器响应后结束计时，以此来计算使用的时间和完成的事务个数，最终以此来估计性能状态。

● PV

PV 即 Page View，意思是页面浏览量或单击率，通常是用来衡量一个网站的主要指标。一个 PV 从狭义上讲等于一个 TPS，可以将页面元素从性能测试结果中分析出来，所以也可以将 PV 和 TPS 整合；但是从广义上来说，一个页面有很多的图片、链接和 CSS 等，而这里的每一个图片、链接都是 PV。

5. 思考时间 (ThinkTime)

思考时间是模拟真实用户在操作过程中的等待时间。从定义上看，ThinkTime 是执行步骤的等待时间，Load Runner 中，就是各个步骤的间隔时间。

6. 迭代 (Iteration)

迭代 (Iteration) 是重复执行过程，性能测试中就是一个事务流程重复执行的过程。通过调整每次迭代的时间，就能控制整个事务流程完成的时间，进而控制 TPS 的大小。

7. 步调 (Pacing)

步调是指两次迭代之间的间隔时间，可以通过设置步调来调整各个 Action 之间的执行等待时间。从定义上看，步调和迭代是绑定在一起的，可以认为是 Iteration Pacing。

8. 集合点 (Rendezvous)

插入集合点是为了衡量加重负载情况下的性能情况。在计划中，可能会要求系统承受 1000 人同时提交数据，在 LoadRunner 中通常在提交数据操作前加入集合点。当虚拟用户运行到提交数据的集合点时，LoadRunner 就会检查同时有多少用户运行到集合点。如果用户人数不到 1000 人，LoadRunner 就会命令到集合点的用户在此等待，当在集合点等待的用户人数到达 1000 人时，LoadRunner 将命令 1000 人同时提交数据，从而达到计划中的需求。集合点是相对的，因为没有绝对的并发，性能测试工具中的集合点并发只是尽可能地在同一时间发送请求，但是服务器是没办法在同一秒接受并处理请求的，就像 CPU 是切换时间片段来进行事务处理一样，看起来像并发处理，实际上是串行机制。

9. 每秒连接数 (Connection per Second)

每秒连接数显示 Web 应用程序在运行过程中每秒建立的 HTTP 连接数。理想情况下，很多 HTTP 请求都应该使用同一个连接也就是使用长连接，而不是每个请求都新打开一个连接。如果程序中不断地打开长连接或者频繁地创建达到上千个短连接，那么应用服务器的响应会越来越慢。通过每秒连接数可以看出服务器的处理情况。

10. 吞吐量 (Throughput)

吞吐量是指单位时间内系统处理的客户请求的数量，直接体现软件系统的性能承载能力。一般来说，吞吐量用字节数/秒 (Bytes/sec) 或页面数/秒 (Hits/sec) 来衡量。从业务的

角度来说,吞吐量也可以用“访问人数/天”或“处理的业务数/小时”等单位来衡量。在大部分性能测试工具的统计中,单击数(Hits)是指客户端发出的 HTTP 请求的数量,而不是指用户在 HTML 页面上的一次单击事件。例如,一次单击事件请求了页面 A,页面 A 包含 3 张图片和一个框架(Frame),则这次单击事件共产生了 5 个单击数(包括对页面 A 本身的请求)。对于交互式应用(Web 应用)来说,吞吐量指标反映的是服务器承受的压力。在容量规划的测试中,吞吐量是一个重点关注的指标,因为它能够说明系统级别的负载能力,一般吞吐量越大,系统单位时间内处理的数据越多,系统的负载能力也越强。

9.1.2 需求分析与策略

需求分为 FRS (Functional Requirement Specification) 和 NFR (Non Functional Requirements),指的是功能性需求和非功能性需求。大部分 NFR 要进行性能测试,主要有负载测试、压力测试和稳定性测试等。

任何一个性能测试项目的开始,都要根据系统性能需求和特性来制定测试方法和策略。在选择合适的测试策略之前,要对性能测试方法的基础理论进行分析,从而帮助理解不同的测试方法。判断一个系统的性能一般取决于以下几个性能指标:

- 系统吞吐量(TPS);
- 系统响应时间;
- 系统资源使用率。

受系统资源的限制,当访问量超过了应用服务的最大容量时,会出现系统缓慢以致阻塞直至崩溃。如图 9-1 所示,最下面的曲线 3 代表请求响应时间;中间的曲线 2 代表吞吐量;最上面的曲线 1 代表资源使用率。

从图 9-1 中可见,在轻负载区(第一条虚线往左的区域)增加负载,应用的响应时间不会有很大变化,但吞吐量和资源利用率还没有达到饱和。在重负载区(两条虚线中间的区域),吞吐量将不再上升,而响应时间开始增加(由于请求排队),造成用户体验下降或者拖慢服务使用方的处理速度。随着负载进一步增加,将进入崩溃区(第二条虚线往右的区域),此时系统资源将主要消耗在资源竞争与调度上,吞吐量反而会开始大幅下降,同时请求队列膨胀,响应时间急剧增加。

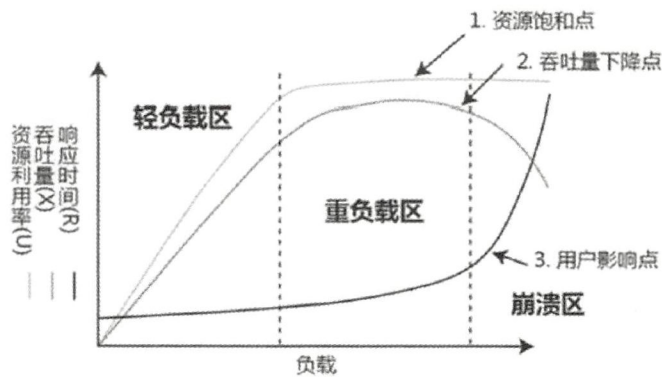


图 9-1 负载曲线

负载控制要尽量做到保证系统在正常情况下工作在轻负载区，寻找最佳并发点，防止进入崩溃区。理想情况下，负载控制要能够将图 9-1 中的曲线变成如图 9-2 所示的曲线。

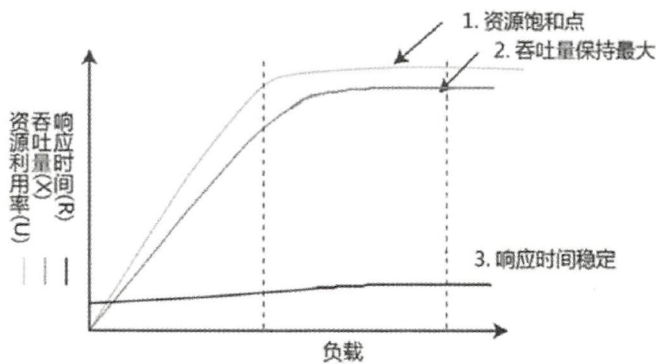


图 9-2 理想负载曲线

1. 测试方法归纳

广义的性能测试根据测试的范围、负载加压模式、数据量和执行时间可以分为性能测试（狭义）、负载测试、压力测试、容量测试、调优测试、稳定性测试和性能基准测试。

性能测试（Performance Testing）是通过对系统增加负载而得出系统的负载能力、性能指标等各种测试行为，包括负载、压力、容量、稳定性和性能指标验证多项性能测试。这是一个概括性的术语，主要衡量系统性能指标，关注点根据不同测试方法而定。如表 8-1 所示列出了各类性能测试方法、测试目的和关注点。

表 9-1 性能测试方法

测试方法	测试方法简述	测试目的和关注点
负载测试 (LoadTesting)	通过模拟系统所承载的并发用户或请求流量的负荷,用不断加压的方式观察系统在不同压力下的性能变化,在满足性能指标的情况下,系统所能承受的最大负载量	通过不同负载测试得出相应性能指标: (1) 得出系统最优 TPS。 (2) 得出系统最优 TPS 时硬件资源利用率。 (3) 得出系统最优并发数
压力测试 (StressTesting)	压力测试是指判断被测系统在强压力下何时出现不可恢复的崩溃现象,如宕机、僵死、异常现象等	着重于系统在强压力下的性能状况: (1) 得出系统崩溃点的 TPS。 (2) 得出系统崩溃点的 TPS 硬件资源利用率。 (3) 得出系统极限并发数
容量测试 (VolumeTesting)	容量测试是为了确定系统可处理同时在线的最大用户数,使系统承受超额的数据容量来发现它是否能够正确处理	(1) 获取系统在拐点时的性能数据。 (2) 获取系统性能点下降时的性能数据 容量测试在资源占用和其他应用方面都有设定的指标,如 CPU、内存、磁盘使用量、平均负载等。这些指标的设定一般采用 80/20 原则,如 CPU 使用率不超过 80%
稳定性测试 (Stability Testing)	稳定性测试是指在给定的负载下,系统长时间健康运行不出现故障、不出现异常的峰值。 (1) 给定的负载一般是以容量极限为参考,用 80/20 原则制定负载模式。但在实际的稳定性测试中,一般都是用恒定的负载进行测试。 (2) 测试执行时间一般按 24 小时 X 7 天。系统用户的业务操作主要发生工作时间内约为 10 小时,所以实际测试时间被设定为 3 天,总计 70 小时。不过有些项目的发布时间紧,测试时间就被设定为 12 个小时	(1) 长时间测试中系统稳定不宕机。 (2) 被测业务场景无性能问题。 (3) 某业务出现错误时,系统可稳定持续运行。关注系统稳定性,要对资源、连接池、JVM 内存变化、TPS、应用系统响应时间和 DB 健康状况进行关注

2. 定义流程

选取性能测试流程时,可以根据公司内部的性能测试流程或者自定义一个性能测试流程,需要多方达成共识。按照如图 9-3 所示的步骤进行性能测试。

- (1) 提交性能测试申请后,收集性能测试需求。
- (2) 建立业务模型,设定测试方案和计划。
- (3) 搭建性能测试环境,配置监控系统并且通过功能测试验证。
- (4) 开始编写性能测试脚本,准备测试数据。
- (5) 设置测试场景并配置监控系统,再执行测试场景。

(6) 执行结束后，收集和分析测试结果。

(7) 最后编写测试报告交给项目组，项目组通过则性能测试结束，不通过则开始调优，调优结束后开始新一轮的测试执行。

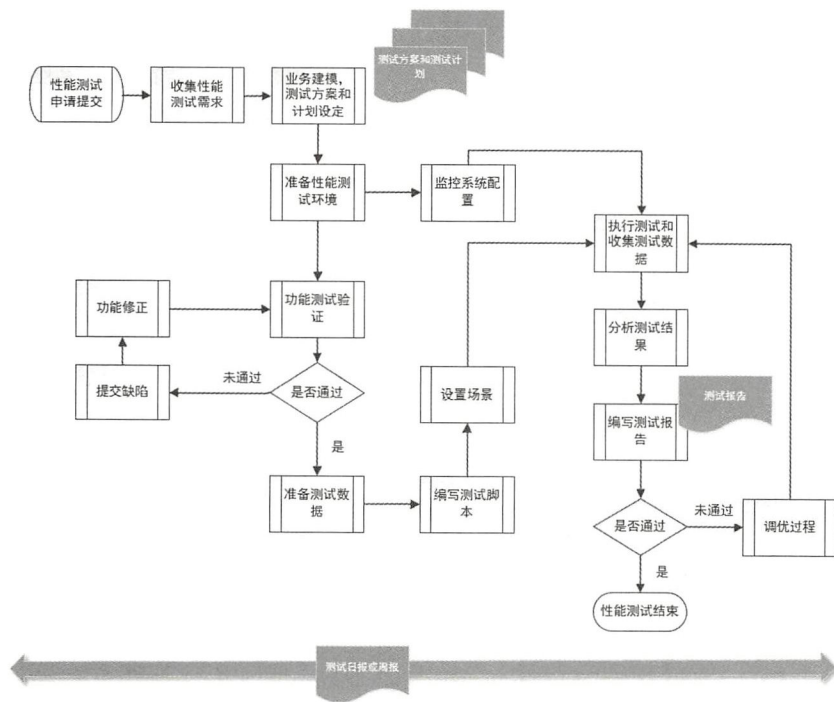


图 9-3 性能测试流程

3. 测试策略

在性能测试项目中，根据不同的产品、系统、架构、通信方式和项目需求，选择不同的测试方法。一般要从如下几个维度来考虑。

首先是通信方式，很多商业性能测试工具包含了众多不同的通信方式测试方法。Load Runner 支持常见的通信协议，如 Web (HTTP/HTML)、CS、ERP/CRM、DB、E-Business、Java、Mailing Service、Wireless、AjaxTruClient、Web Service 和 Citrix 等。

其次是被测系统环境结构，如单系统和集成系统环境。

最关键的是需求和项目类型、测试时间限制等。



4. 选取性能测试工具

当一个项目选定了测试策略，这时候需要的是选取一款合适的性能测试工具。通过性能测试工具能够准备测试脚本、执行性能测试、进行收集和分析在性能测试过程中获得的数据，帮助测试人员高效地完成性能测试工作。两款常用的性能测试工具是 Load Runner 和 JMeter。

9.2 测试利器之 LoadRunner

LoadRunner 是 HP 开发维护的预测系统行为和性能的工业标准级商业负载测试工具。通过模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题，它能够对整个架构进行性能测试。通过使用 LoadRunner，能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

9.2.1 LoadRunner 安装贴士

LoadRunner 组件分为 4 个应用，如图 9-4 所示。

- (1) Virtual User Generator: 称为 VuGen，可以录制、编写和调试脚本。
- (2) Controller: 设计、执行测试场景和监控服务器数据。
- (3) Analysis: 用来收集测试结果，生成分析报告。
- (4) LoadRunner Agent: 用来作为执行性能测试的负载机。

组件中的 Virtual User Generator、Analysis、Agent 是不需要序列号的，但是 Controller 需要根据不同的协议和支持的并发用户数进行收费。早期 Load Runner 版本中的独立控制器 (Standalone Controller) 被现在的 HP ALM Performance Center 替代，按测试项目分类管理、上传脚本、设计场景和执行测试场景，多个测试人员可以协同工作，合理分配性能测试时间，预订负载机的时间片，还可以长久保存测试结果。

LoadRunner 的安装相对比较简单，这里需要注意以下 3 点：

- (1) 尽量安装在默认目录下，安装目录不要带有中文。

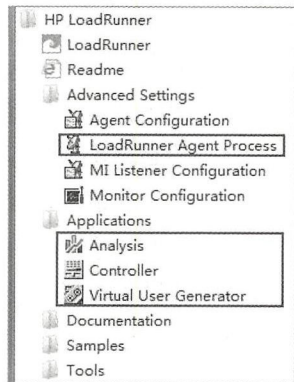


图 9-4 Load Runner 组件



(2) 安装 Controller 的机器需要较高配置的 CPU 和内存，建议 CPU 频率 2 GHz 以上，内存容量 4GB 以上。

(3) 由于 LoadRunner Agent 是用来作为压力测试的负载机模拟并发用户的，其安装的机器也需要高配置的 CPU 和内存，建议 CPU 频率 2 GHz 以上，内存容量 4GB 以上。

提示：对网络带宽有要求的性能测试，可以使用 LoadRunner 的模拟宽带资源功能，在 RuntimeSettings 中可以进行网络带宽模拟，如图 9-5 所示。

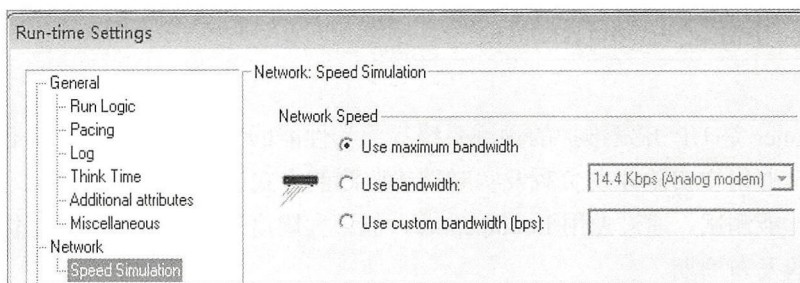


图 9-5 LoadRunner 网络带宽模拟

9.2.2 脚本与优化

使用 Load Runner 的第一步是准备脚本，下面主要介绍 Web (HTTP/HTML)、Web Services 和 Java Vuser 协议。在脚本录制后，还需要优化脚本以增加脚本的健壮性与通用性。以下是准备脚本的常见步骤。

(1) 新建脚本并选择协议：选择合适的协议新建脚本。

(2) 录制脚本或者编写脚本：类似 Web (HTTP/HTML) 协议的脚本可以录制用户在浏览器上操作的请求，而有些 Java 协议的脚本却只能自己编写脚本，不可录制。

(3) 添加思考时间和事务名称：添加思考时间使得脚本运行时更接近实际用户的操作，事务名称的设置可以对请求分组，更好地计算事务的响应时间。

(4) 添加检查点：脚本是否正确运行需要添加检查点。

(5) 关联脚本：根据关联规则，通过关联函数定义左右边界值，在服务器所响应的内容中获得所需的返回值，以变量的形式替换录制时的静态值，从而向服务器发出正确的后续请求。

(6) 参数化脚本：对于多样化的测试数据进行参数化，脚本执行时可以选择不同的测试数据。



(7) 调试脚本：打开脚本日志，跟踪输出信息，对比树形窗口的录制和播放的响应内容，确保脚本的准确性。

1. Web (HTTP/HTML) 协议

首先介绍如何创建 Web (HTTP/HTML) 协议的脚本，参照以下步骤操作即可。Web (HTTP/HTML) 协议主要以 HTTP 请求作为录制基础，而 Web (Click and Script) 是监控用户单击页面的动作，不要将这两种协议搞混。

(1) 打开 Virtual User Generator 应用后，单击“New”按钮可以新建脚本，如图 9-6 所示，也可以选择菜单项“File”→“New”，或者按快捷键 Ctrl+N。

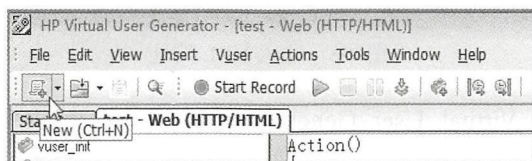


图 9-6 新建脚本

(2) 打开“New Virtual User”对话框，选择协议“Web (HTTP/HTML)”，如图 9-7 所示，然后单击“Create”按钮。

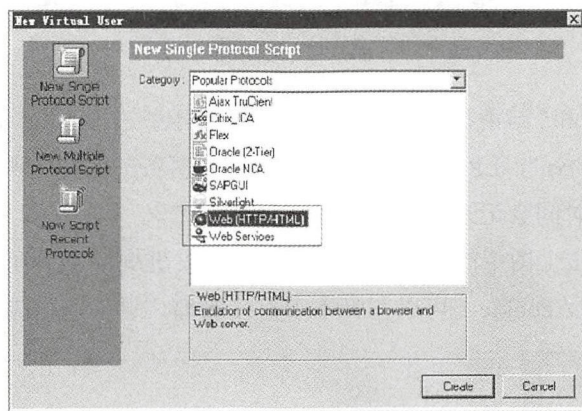


图 9-7 Load Runner Web 协议

(3) 在“Start Recording”对话框的文本框“Program to record”中，选择浏览器（默认支持至 Firefox3.6 或 IE6），如图 9-8 所示，若打上 LoadRunner 的补丁也可以支持 IE8 或者 Firefox 更高版本。然后输入 URL 地址和 Workingdirectory，单击“OK”按钮开始录制脚本。



- URL Address: 需要录制脚本的 URL 地址。
- Working directory: 录制脚本的存放目录。

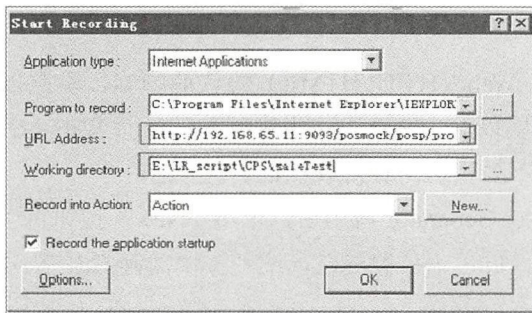



图 9-8 输入 URL 地址

(4) 录制脚本时，Virtual User Generator 会启动一个代理的浏览器，并显示这个“Recording...”录制框，可以停止或者暂停录制。每个操作前后都可以添加 Transaction，如图 9-9 所示的方形框中的两个时钟箭头标记。在录制脚本时，每个操作可以根据 Transaction 分开，这样不至于造成后期脚本维护困难。



图 9-9 录制框

添加“开始事务”和“结束事务”后，在脚本代码中会产生语句 `lr_start_transaction("txn")` 和 `lr_end_transaction("txn", LR_AUTO)`，其中“txn”是事务的名称。两者之间的所有请求在执行后，产生的响应时间会记录在“txn”下。

录制完成后，单击上图中的蓝色方形的“停止”按钮，可以看到类似的脚本，如图 9-10 所示。如果不需要 cookie，可以在脚本中直接删掉，这不影响脚本的正常流程。

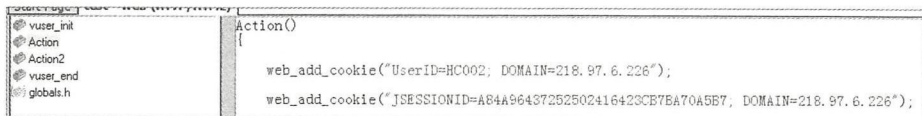


图 9-10 录制后显示的 Web 脚本样例

2. Web Services 协议

Web Services 在 Load Runner 中有两种脚本方式：Web 协议和 Web Services 协议。Web



协议脚本的录制如上节所述,这一节介绍如何新建 Web Services 脚本。Web Services 协议可以添加 Web Services 的调用方法和参数,也可以导入 SOAP 请求,分别为 `web_service_call()` 和 `soap_request()`。本节中所使用的实例是一个公开的天气服务: <http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl>。

(1) 打开 Virtual User Generator, 单击工具栏处的“New”按钮, 新建脚本, 并选择 Web Services 协议, 如图 9-11 所示, 然后单击“Create”按钮。

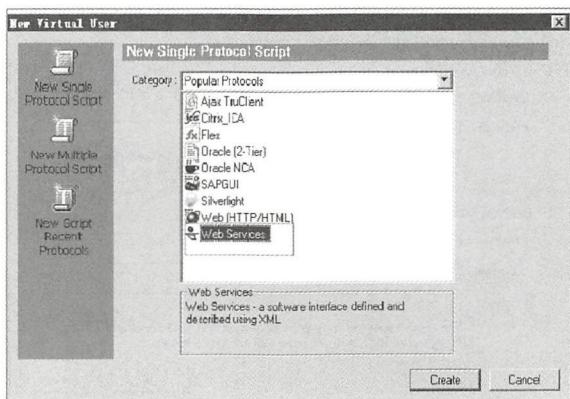


图 9-11 选择“Web Services”协议

(2) 在 Virtual User Generator 中, 单击“Manager Services”按钮, 如图 9-12 所示。

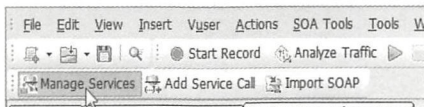


图 9-12 单击“Manager Services”按钮

(3) 在弹出的“Manager Services”窗口中, 单击“Import”按钮, 在打开的对话框中输入 Web Services 需要请求的 URL。注意要在 WebServices 地址后面加上“?wsdl”, 如图 9-13 所示。

如果 WSDL 需要用户名和密码, 可以单击“Connection Settings...”按钮, 打开连接设置窗口, 输入连接用户名和密码, 如图 9-14 所示。

(4) 在“Manager Services”窗口中, 输入对应的地址和连接设置后, 单击“Import”按钮后, 可以看到对应的 Web Services 信息, 如图 9-15 所示。

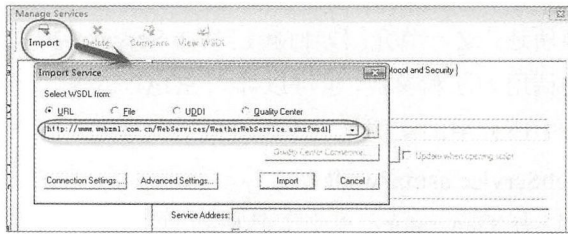


图 9-13 输入 Web Services 的 URL 地址

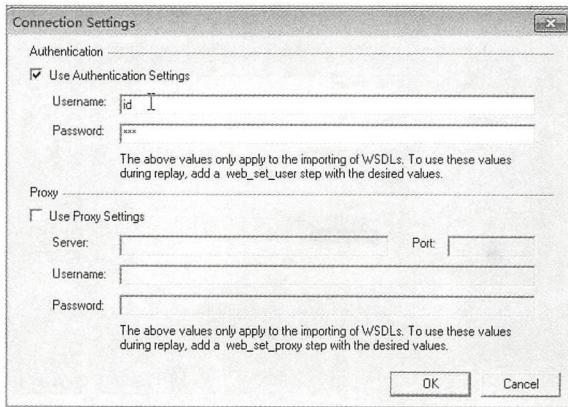


图 9-14 输入 Web Services 连接用户名和密码

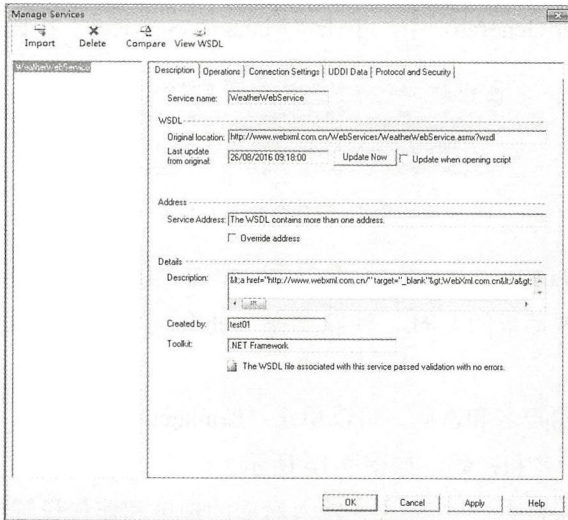


图 9-15 查看 Web Services 信息



(5) 返回 Virtual User Generator 应用的主界面，单击“Add Service Call”按钮，如图 9-16 所示，可以添加新的服务调用。

(6) 打开“New Web Service Call”对话框，可以选择需要添加的“Service”和“Operation”，如图 9-17 所示。根据需求，选择输入参数（Input Arguments）和输出参数（Output Arguments）。

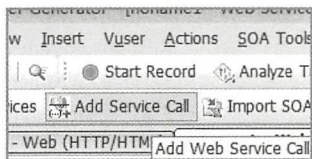


图 9-16 单击“Add Service Call”按钮

图中的例子展示出 WeatherService 服务下的 getAutomaticMoreWeather 操作。这个操作有 ip 和 UserId 两个输入参数。选择参数 ip，在窗口右边区域内可以修改参数的输入类别。输入参数有 3 个类型。

- Nil: 空值。
- Value: 来自于参数表。
- Generate auto-value for this argument: 自动产生值。

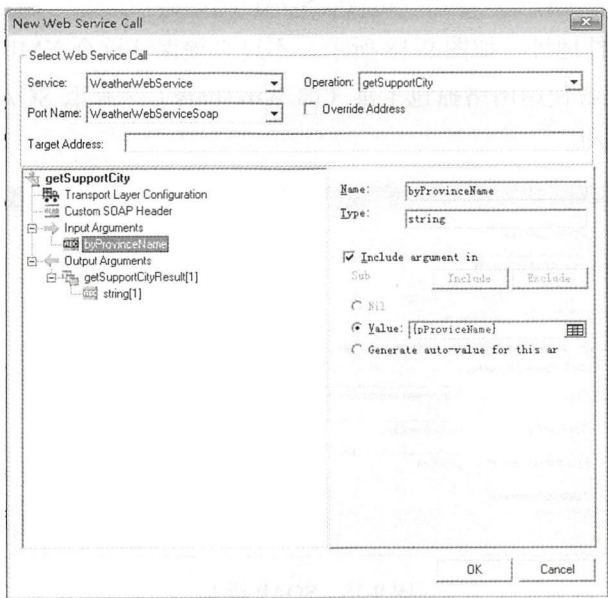


图 9-17 “New Web Service Call”对话框

(7) 单击“OK”按钮，生成的代码在 web_service_call 方法中，也可以此时参数化。在如下代码中，Service Call 函数 getWeatherbyCityName 的输入值接收参数 pProvinceName，输出值返回至参数 pOutCityName 中。



```
web_service_call( "StepName=getSupportCity_101",
    "SOAPMethod=WeatherWebService|WeatherWebServiceSoap|getSupportCity",
    "ResponseParam=response",
    "Service=WeatherWebService",
    "ExpectedResponse=SoapResult",
    "Snapshot=t1472175216.inf",
    BEGIN_ARGUMENTS,
    "byProvinceName={pProvinceName}",
    END_ARGUMENTS,
    BEGIN_RESULT,
    "getSupportCityResult/*[1]=pOutCityName",
    END_RESULT,
    LAST);
```

(8) 除了使用 Manage Services 方式外，还可以使用 Import SOAP XML 请求。在 Virtual User Generator 应用主界面中，单击“Import SOAP”按钮，打开“Import SOAP”对话框，选择 XML 请求的文件地址，如图 9-18 所示。不过必须保证这个 XML 请求在 SOAP UI 软件中能发送成功，或者使用网络抓包工具（如 Wireshark）来抓取 SOAP Body 的内容，直接保存成 XML 并导入。

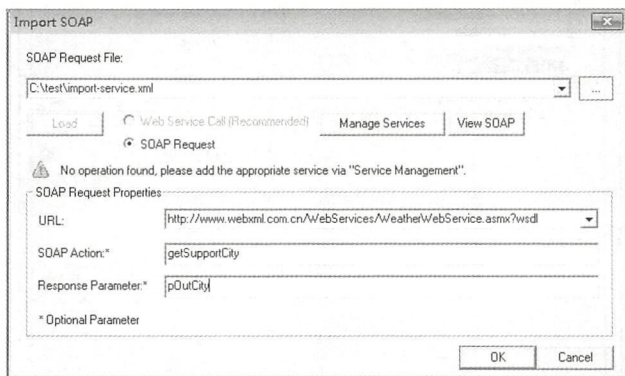


图 9-18 SOAP 请求

导入成功后，会生成 soap_request 函数，代码如下：

```
soap_request("StepName=SOAP Request",
    "URL=http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl",
    "SOAPEnvelope="
    "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\"
```



```

xmlns:web="\http://WebXml.com.cn/">
    "<soapenv:Header></soapenv:Header>"
    "<soapenv:Body>"
        "<web:getSupportCity>"
            "<web:byProvinceName></web:byProvinceName>"
        "</web:getSupportCity>"
    "</soapenv:Body>"
"</soapenv:Envelope>",
"SOAPAction=getSupportCity",
"ResponseParam=response",
"Snapshot=t1472176170.inf",
LAST);

```

此时，还需要添加 SOAP Header 内容，否则会抛出错误“Error: The reason for the SOAP fault is:“服务器未能识别 HTTP 头 SOAPAction 的值:getSupportCity。””。

```

web_add_header("POST",
    "http://www.webxml.com.cn/WebServices/WeatherWebService.asmx HTTP/1.1");
web_add_header("Content-Type",
"application/soap+xml;charset=UTF-8;action=\http://WebXml.com.cn/getSupportCity\"");
web_add_header("Host",
    "www.webxml.com.cn");

```

这里还有一种方法，可以手写 web_custom_request。但是 SOAPXML 中的所有字段要自己填。如果遇到很多参数，建议还是使用上面的方法。

```

web_add_header("Content-Type",
    "application/soap+xml;charset=UTF-8;");
web_reg_save_param_ex(
    "ParamName=pCityNames",
    "LB=",
    "RB=",
    "Ordinal=1",
    SEARCH_FILTERS,
    "Scope=BODY",
    LAST);
web_custom_request("SoapHTTPRequest",
    "URL=http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl",
    "Method=POST",
    "Mode=HTTP",
    "Body=<?xml version='1.0' encoding='utf-8'?>"
    "<soap12:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance\" "

```

质量全面管控：从项目管理到容灾测试

```
"xmlns:xsd="http://www.w3.org/2001/XMLSchema" "
"xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  "<soap12:Body>"
  "<getSupportCity xmlns="http://WebXml.com.cn/">"
  "<byProvinceName></byProvinceName>"
  "</getSupportCity>"
  "</soap12:Body>"
  "</soap12:Envelope>",
LAST);
```

(9) 切换到 TREE 界面，查看定义的函数请求和服务器的返回值。

3. Java Vuser 协议

Load Runner 可以创建 Java Vuser 协议的脚本，支持对 Java 应用程序的接口测试。本节讲述 Java Vuser 协议脚本的创建。

(1) 在 Virtual User Generator 界面中，单击工具栏处的“New”按钮，新建脚本并选择 Java Vuser 协议，如图 9-19 所示，然后单击“Create”按钮。

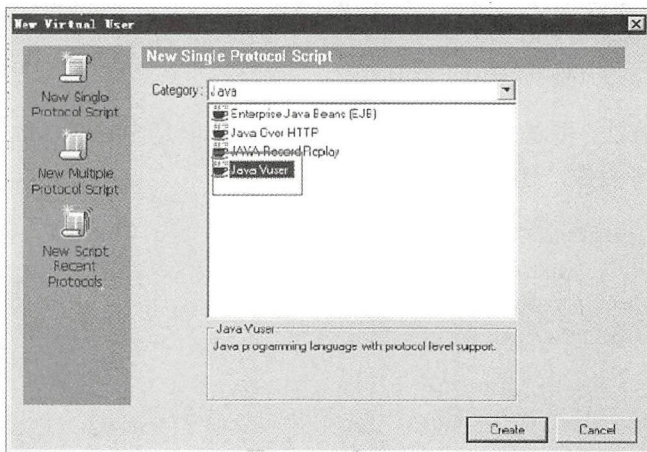


图 9-19 选择“Java Vuser”协议

(2) 配置 JDK 路径并添加 Jar 路径（如果调试脚本不报相关错误，这一步可以忽略），如图 9-20 所示，然后单击“OK”按钮。

(3) Java Vuser 脚本需要引入 lrapi.lr 类，其执行顺序是：init()→action()→end()，其中的 action()方法会被重复执行，具体写法如图 9-21 所示。

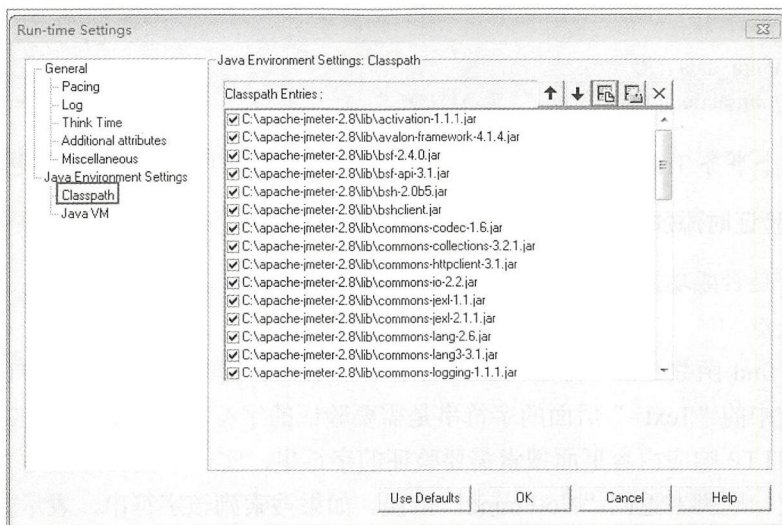


图 9-20 配置 JDK 路径并添加 jar 路径

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.log4j.Logger;

import lrapi.lr;

public class Actions
{
    public int init() throws Throwable {
        return 0;
    } //end of init

    public int action() throws Throwable {
        return 0;
    } //end of action

    public int end() throws Throwable {
        return 0;
    } //end of end
}
```

图 9-21 引入 lrapi.lr 类

4. 添加思考时间和事务名称

按照前面介绍的内容，录制完脚本后，还需要优化脚本才能保证脚本运行正常。在每个事务前后，添加 Thinktime（思考时间）和 Transaction（事务名称）。实例中的 THINK_TIME 需要填写具体的思考时间，如 lr_think_time(5); 表示等待 5 秒后，继续执行后续步骤。

```
lr_think_time(THINK_TIME);
lr_start_transaction("trade_create");
```


质量全面管控：从项目管理到容灾测试

```
web_url(*****);
web_submit_data(*****);
lr_end_transaction("trade_create",LR_AUTO);
```

提示：每个事务 Transaction 都是一个操作，不是一个 HTTP 请求，经常包含很多请求。

5. 脚本验证时添加检查点

脚本运行是否成功，需要添加检查点或关联参数来验证。

● 检查点

web_reg_find 函数是一个注册函数，对下一个 HTTP 提交的请求验证 HTTP 响应。它的第二个参数中的“Text=”后面的字符串是需要验证的字符串。当下一个 HTTP 请求发送后，在它的 HTTP 响应内容里面搜索需要验证的字符串，实例中验证的值是“Welcome”。如果搜索失败，则脚本返回“NotFound”错误。如果搜索到该字符串，表示验证成功，继续发送后面的请求。

```
web_reg_find("Fail=NotFound", "Text=Welcome", LAST);
```

● 关联函数

关联函数是用来抓取需要验证的字符串的左右边界，如果找不到关联的返回值，那么表示取值失败，所以也可以作为脚本验证。

```
web_reg_save_param("payList", "LB=outboundFlight value=", "RB= checked >", LAST);
```

6. 关联技术

Web HTTP 请求中很多参数都是动态的，也就是 LoadRunner 中的关联（Correlation）。在脚本回放过程中，客户端发出请求，通过关联函数所定义的左右边界值（也就是关联规则），在服务器响应的内容中查找并得到相应的值，该值以变量的形式替换录制时的静态值，从而向服务器发出正确的请求，这种动态获得服务器响应内容的方法被称做关联。例如：用户每次登录系统时会产生一个新的会话，而这个会话获取不正确会导致下一个请求（比如查看订单的请求）发送不成功。这时“会话”的值就需要进行关联，从登录请求返回的响应内容中获取到正确的值，然后作为参数传给查看订单请求，最终查看订单返回正确结果。

关联技术分为自动关联和手动关联：

● 自动关联

自动关联包含两种机制。一种是 LoadRunner 通过对比录制和回放时服务器响应的不同，提示用户是否进行关联。用户可以自己创建关联规则，这个功能可以方便用户获得需

要关联的部分，但同时也存在一定的问题，如自动关联所检测到的关联点不一定真的需要进行关联，这就要根据实际情况进行判断；有些需要关联的动态数据自动关联无法找到，这就需要手动关联。另一种是 LoadRunner 自带的自动关联规则，在录制脚本时，会根据这些规则自动创建关联。

● 手动关联

手动关联的大致步骤是：首先录制测试脚本，同样的步骤和输入参数录制两遍，使用 WinDiff 或 BeCompare 等文本比较工具找出两次脚本的不同，判断是否需要进行关联确定插入关联的位置，然后手动在查看树（通过菜单项“View”→“Tree View”打开）中使用 web_reg_save_param 函数手动建立关联，将脚本中用到关联的数据用参数代替，最后验证关联的正确性。

例如：如图 9-22 所示，上面一排表格的 HTTP View 显示的是录制的 Request 和 Response，下面一排表格是回放的 Request 和 Response。在右侧的 Response 区域内，可以看到 JSESSIONID 在录制和回放时发生变化，如果脚本后续的请求依赖 JSESSIONID 值，那么 JSESSIONID 值就需要做关联。

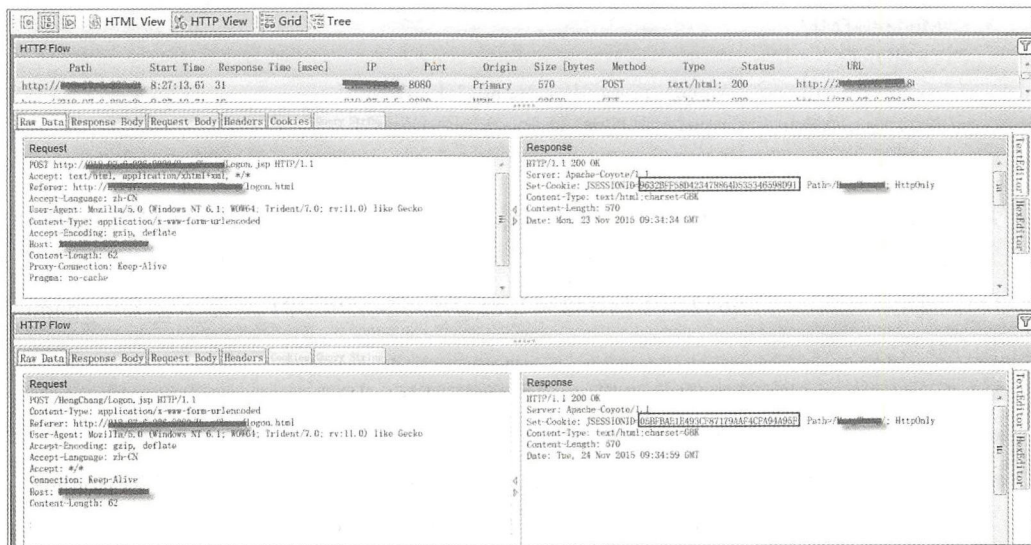


图 9-22 手动关联对比

动态参数一般都是从前一个请求的响应内容中获取的，这里要用到关联函数 web_reg_save_param 或 web_reg_save_param_ex。这两个关联函数都是注册函数，下一个请求获得服务器响应后，在响应内容中搜索字符串，并保存到参数变量中。

质量全面管控：从项目管理到容灾测试

```

web_reg_save_param_ex("ParamName= Corr_JSessionID",
    "LB=JSESSIONID=",
    "RB= Path=",
    SEARCH_FILTERS,
    "Scope=Headers",
    LAST);
web_url("Redirector", "URL=http://10.108.6.226:8080/test/Redirector?ComponentURL=/Main.jsp",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t3.inf",
    "Mode=HTML",
    LAST);
web_submit_data("j_spring_security_check",
    "Action=http://10.108.6.226:8080/app/j_spring_security_check",
    "Method=POST",
    "TargetFrame=",
    "RecContentType=text/html",
    "Referer=http://{pTestEnv}/app/login;jsessionid={Corr_JSessionID}",
    "Snapshot=t2.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=j_username", "Value={pLoginUserName}", ENDITEM,
    "Name=j_password", "Value={pLoginUserPwd}", ENDITEM,
    LAST);


```

实例中的 `web_url("Redirector", "URL=xxxx")` 从客户端发起请求，获得服务器响应。然后，`web_reg_save_param_ex` 函数根据左右边界，搜索定位到 `JSESSIONID` 的实际值，即图 9-22 中圈出的一串字符 **9632BFF58D423478864D535346598D91**。`web_reg_save_param_ex` 函数中的参数含义如下。

- `LB=`指待搜索字符串的左边界，这里指 `JSESSIONID=`。
- `RB=`指待搜索字符串的右边界，这里指 `Path=`。
- `SEARCH_FILTERS` 区域下的 `Scope=` 设置了响应的搜索范围，可以是 `Headers`（响应头）、`Body`（响应主体）、`All`（所有的响应内容）。
- 关联函数保存的参数变量名为 `Corr_JSessionID`，获取返回值后，被作为下一个请求（`web_submit_data("j_spring_security_check")...`）的一部分，提交给服务器。

7. 参数化技术

Web HTTP 请求需要使用不同的测试数据来模拟不同的用户、账户样本等，保证数据的

多样化,使性能测试的结果更贴近真实情况。在 LoadRunner 中,这些数据可以从文本文件、LoadRunner 自置带的函数或者数据库中读取,也可以通过配置参数来实现。在 Virtual User Generator 的工具栏中,单击高亮的  按钮,如图 9-23 所示。

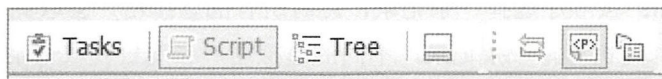


图 9-23 参数化工具栏

在打开的对话框中,单击“New”按钮新建一个参数 pCustPRCID,在左侧列表中输入参数名“pCustPRCID”,右侧的参数属性设定 Parameter type (参数类型)为 File,参数的数据文件为 pCustPRCID.dat,设置完成后如图 9-24 所示。其中“Add Column...”按钮可以新增数据列,“Add Row...”按钮可以新增数据行,以便直接在列表框中增加数据;也可单击“Edit with Notepad”按钮,直接在 pCustPRCID.dat 文件中修改添加数据。

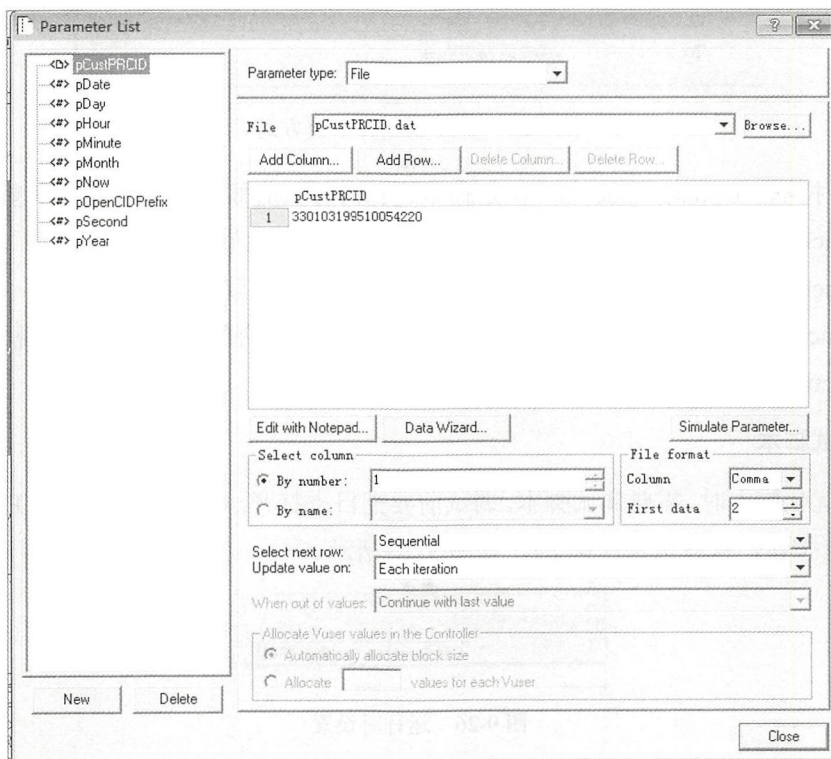


图 9-24 新建一个参数

质量全面管控：从项目管理到容灾测试

参数化类型有很多种，读者可以参考 LoadRunner 自带的帮助文档。在选择行数据时，要注意参数的调用方式。

(1) 下拉框“Select next row”有 3 种方式选择参数文件中的下一行数据。

- Sequential: 按顺序选择一条记录，此时所有的虚拟用户会取到同一个值。
- Random: 随机选择一条记录，不同的虚拟用户会随机获取记录，值可能不一样。
- Unique: 选择不重复的一条记录。在选择这一方式时，会出现一个子选项，当参数值遍历完后没有剩余值时，按照其中的规则来处理，如图 9-25 所示，规则包括 Abort Vuser（停止虚拟用户，脚本会不再执行）、Continue in a cyclic manner（取值从头开始遍历一遍）和 Continue with last value（按最后一个值来获取）。

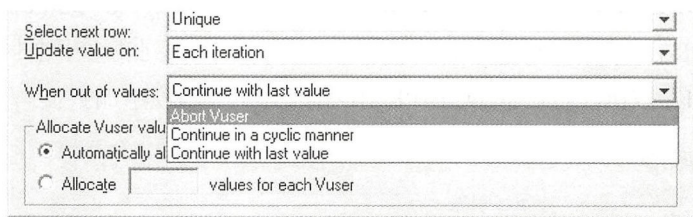


图 9-25 参数 Unique 取值方式

(2) 下拉框“Update value on”决定脚本运行时什么时候开始选取下一条参数数据。

- Each iteration: 每一个 Action 循环，会取一次新的值。
- Each occurrence: 每一次使用到这个参数时，会获取一次新的值。
- Once: 只在第一次使用到这个参数时，会获取一次新的值，以后出现同一参数或 Action 循环调用时都不会修改值。

8. 调试脚本

(1) 在优化脚本时，需要调试脚本。调试前要把日志打开。单击工具栏中的“Edit Runtime Settings”按钮，或者按快捷键 F4，如图 9-26 所示。



图 9-26 运行时设置

(2) 在打开的对话框的左侧列表框中选择“Log”选项，在右侧选项区中勾选“Enable

logging”复选框（启动日志），再选择“Always send messages”单选项，就可以开启日志查看级别了，如图 9-27 所示。

- Standard log（常规日志）：在脚本执行时，发送常规的函数或信息的子集给日志。
- Extended log（扩展日志）：在脚本执行时，发送详细的函数或信息给日志。这里可以选择 3 个复选框：Parameter substitution（日志返回脚本运行时使用的参数和值）、Data returned by server（日志返回服务器的所有信息，像 http response header 和 body）和 Advanced trace（日志返回所有虚拟用户的信息和函数调用）。

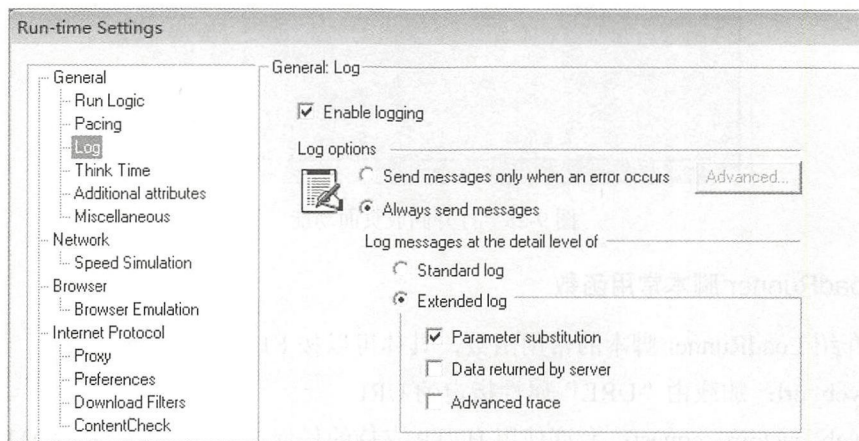


图 9-27 “Run-time Settings”对话框

在调试脚本时，日志级别的设定很有讲究。返回日志越多，运行的时间越长。所以在初步调试时，打开 Parameter substitution 即可，足够判断参数传递是否正确。在此基础上，如果还找寻不出原因，再打开 Data returned by server 来更进一步地分析返回的 Response 内容。

提示：在性能测试时，需要关闭日志，否则大量的日志文件会占用 Controller 机器大量的内存。

(3) 如果是 Web HTTP 请求，在调试脚本时，还可以打开回放页面功能。操作步骤是：选择菜单项“Tools”→“General Options”，在打开的“General Options”对话框的 Display 选项卡中，勾选“Show run-time viewer during replay”复选框，如图 9-28 所示。设置后，脚本回放时会显示页面的执行过程。

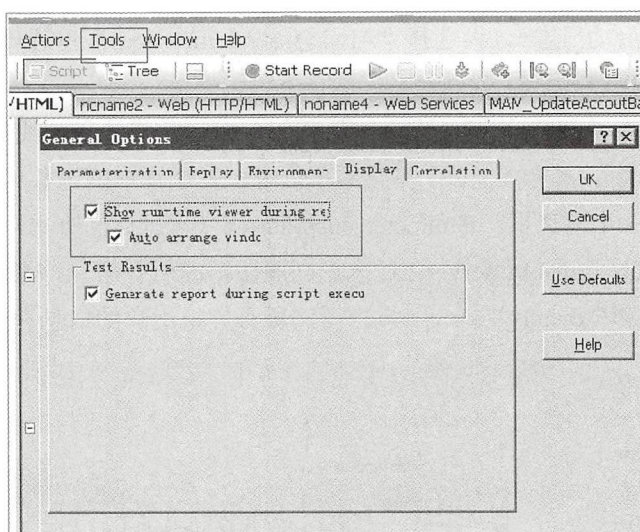


图 9-28 打开回放页面功能

9. LoadRunner 脚本常用函数

下面介绍 LoadRunner 脚本的常用函数，具体可以按 F1 键查看帮助。

- web_url: 加载由“URL”属性指定的 URL。
- web_custom_request: 允许使用 HTTP 支持的任何方法来创建自定义 HTTP 请求。
- web_submit_data: 执行“无条件”或“无上下文”的表单。
- web_submit_form: 模拟表单的提交。
- web_image: 在定义的图像上模拟鼠标单击。
- web_link: 在定义的文本链接上模拟鼠标单击。
- web_reg_save_param("参数名","LB=左边界","RB=右边界",LAST): 注册函数，在参数值出现前使用，注册成功时返回值为 0，注册失败时返回值为 1。左右边界需根据 TreeView 里的 SeverResponse 内容来确定，能获取第一个符合条件的数值。
- web_reg_save_param("参数名","LB=左边界","RB=右边界","Ord=All",LAST): 当参数有多个值时，加上"Ord=All"后可获取所有的数值。注册成功后，{参数名_count}表示取得的数值个数，{参数名_1}为第一个数值，{参数名_2}为第二个数值。
- web_reg_find: 在后面的 HTTP 请求中注册对 HTML 源或原始缓冲区中文本字符串的搜索。

- `web_global_verification`: 在所有后面的 HTTP 请求中搜索文本字符串。
- `web_image_check`: 验证指定的图像是否存在于 HTML 页内。
- `lr_save_string`: 将字符变量里的值传递给指定参数。
- `lr_eval_string`: 取得参数的数值。
- `lr_start_transaction`: 标记事务的开始。
- `lr_end_transaction`: 标记事务的结束。
- `lr_stop_transaction`: 停止事务数据的收集。
- `lr_wasted_time`: 消除所有打开事务浪费的时间。
- `lr_debug_message`: 将调试消息发送到输出窗口。
- `lr_error_message`: 将错误消息发送到输出窗口。
- `lr_log_message`: 将输出消息直接发送到 `output.txt` 文件, 文件位于 Vuser 脚本目录中。
- `lr_output_message`: 将消息发送到输出窗口。
- `lr_message`: 将消息发送到 Vuser 日志和输出窗口。
- `web_set_certificate`: 使 Vuser 使用在 Internet Explorer 注册表中列出的特定证书。
- `web_set_user`: 指定 Web 服务器的登录字符串和密码, 验证用户身份的区域。
- `web_concurrent_end`: 标记并发组的结束。
- `web_concurrent_start`: 标记并发组的开始。
- `web_add_cookie`: 添加新的 Cookie 或修改现有的 Cookie。
- `web_remove_cookie`: 删除指定的 Cookie。
- `lr_xml_get_values()`: 获取查询到的 XML 元素的值。
- `lr_xml_set_values()`: 设置查询到的 XML 元素的值。
- `lr_xml_find()`: 验证查询到需要的 XML 值。

9.2.3 设置场景

打开 HP LoadRunner Controller 程序, 选择菜单“File”下的子菜单项“New”, 或按 Ctrl+N 组合键, 打开新建场景界面, 导入准备好的 LoadRunner 脚本, 如图 9-29 所示, 然

后设置执行方式。

- Initialize: 初始化 Vuser 的方式。
- Start Vusers: 启动 Vuser 的方式和总的并发用户数，默认为按每 15 秒启动 2 人，直至达到 10 个用户。
- Duration: 测试执行持续的时间，此处为 1 个小时。
- Stop Vusers: 停止 Vuser 的方式，默认为按每 30 秒停止 5 人。

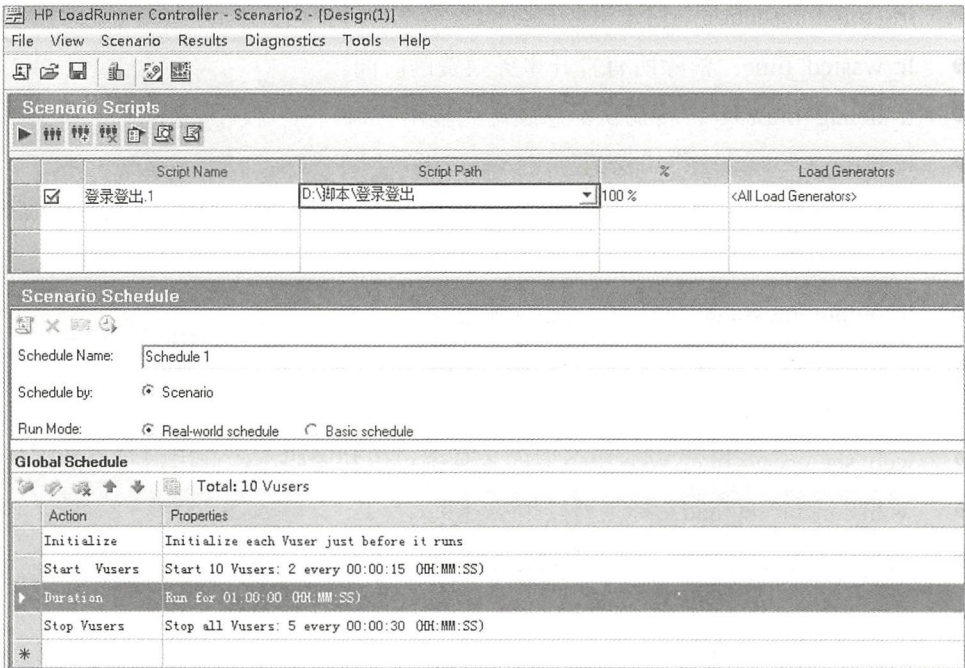


图 9-29 创建场景并导入脚本

9.2.4 运行场景

场景创建完成后，开始运行场景。选择菜单“Results”，设置测试结果保存的路径，然后在主界面下方选择“Run”选项卡，切换到场景运行界面。单击界面正中间的“Start Scenario”按钮，开始运行场景。运行后，“Start Scenario”按钮变为灰色，如图 9-30 所示。

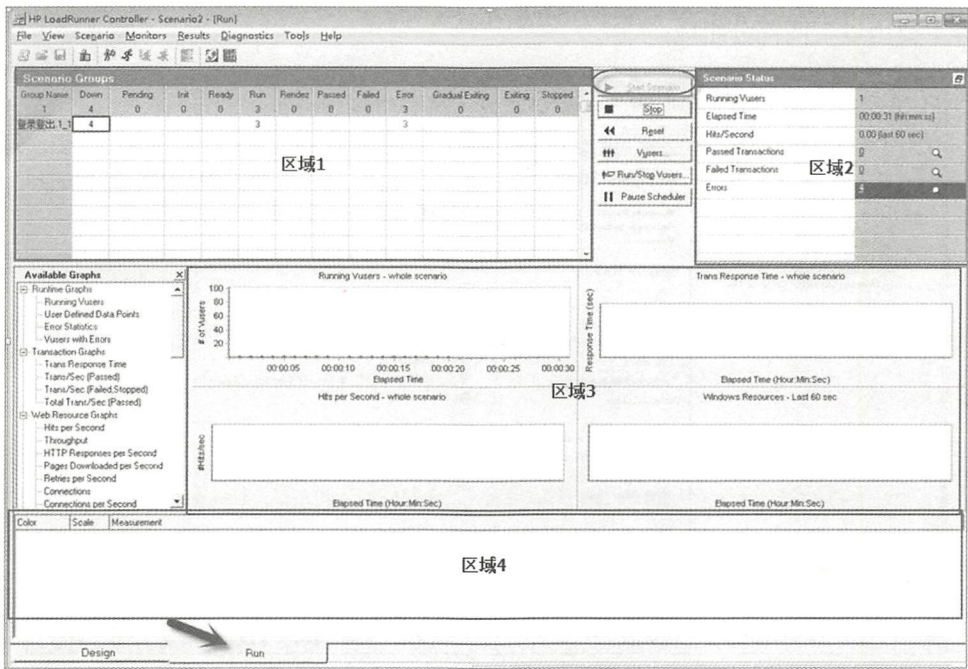


图 9-30 运行场景

下面简单解释一下场景运行界面的各个区域功能。

(1) 左上方区域 1 是“Scenario Groups”区域，其中列出了每个测试脚本的虚拟用户运行状态。例如，在图 9-30 中，有 3 个用户在运行中，3 个用户运行错误，4 个用户还未开始启动。

(2) 右上方区域 2 是“Scenario Status”区域，其中列出了 Running Vuser（运行中的虚拟用户数）、Passed Transaction（通过的事务数）、Failed Transaction（失败的事务数）和 Errors（错误数）等。

(3) 中间的区域 3 列出了 Running Vuser、Response Time、Hits per Second 和 Windows Resources 图表，也可以根据需要自行添加或替换需要监视的图表。

(4) 选中其中一个图表，会在最下面的区域 4 中显示图表的具体数据。

9.2.5 收集和分析结果

使用 LoadRunner Analysis 程序可以打开 LoadRunner 的测试结果，如图 9-31 所示。测

试报告主要由以下两部分组成。

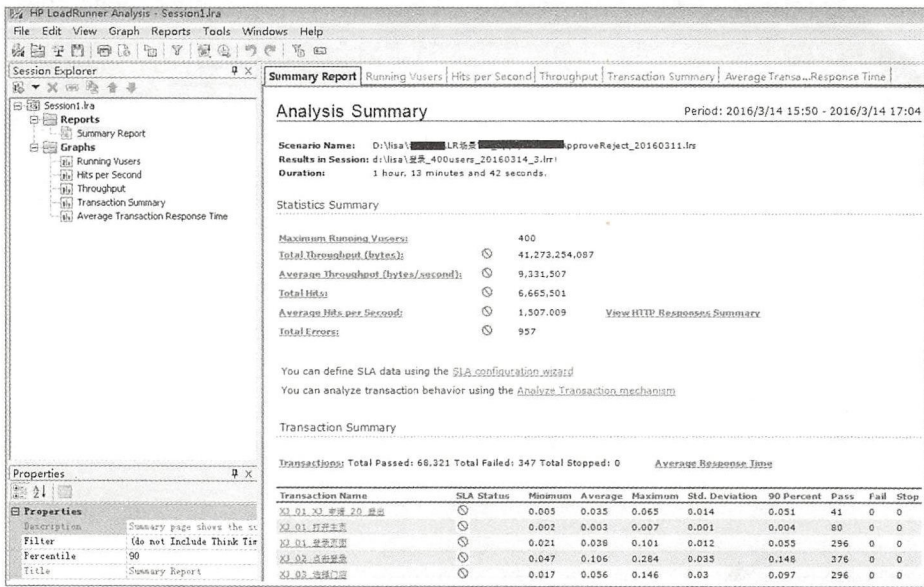


图 9-31 Analysis 测试结果

(1) Summary Report 汇总报告

报告汇总包括执行场景名、执行时间、持续时间、统计汇总和交易汇总。

- Statistics Summary 罗列了运行 Vuser 并发总数、吞吐量、单击率和错误数等。
- Transaction Summary 列出了各事务的响应时间，包括 Min、Avg、Max、Std.、90% 响应时间、事务通过数、事务失败数和错误数等。其中，各列含义如下。
 - “Transaction Name” 显示事务的名称。
 - “SLA Status” 显示事务是否达到 SLA 的标准设定，图例中由于没有设定 SLA，所以显示的是禁用状态图标。
 - “Minimum” 是事务的最小响应时间，以秒为单位。
 - “Average” 是事务的平均响应时间，以秒为单位。
 - “Maximum” 是事务的最大响应时间，以秒为单位。
 - “Std. Deviation” 是事务的响应时间的标准方差。
 - “90 Percent” 是事务按响应时间排序，90% 的事务小于的响应时间。
 - “Pass” 是事务的通过数量。

- “Fail” 是事务的失败数量。
- “Stop” 是事务中断未执行的数量。

(2) Graphs 图表

默认结果包含 5 个图表：Running Vusers（运行时的虚拟用户数）、Hits per second（单击率）、Throughput（吞吐量）、Transaction Summary（事务汇总）和 Average Transaction Response Time（平均事务响应时间）。可以根据需要，添加更多的图表，过滤显示的时间。

9.3 测试利器之 JMeter

JMeter 是 Apache 组织开发的基于 Java 的性能测试工具。JMeter 可以用于对服务器、网络或对象模拟巨大的负载，在不同压力类别下测试其强度和分析整体性能。它和 Load Runner 一样可以实现分布式负载测试。针对 Sockets 协议、SOAP 协议，JMeter 要比 Load Runner 更加容易使用。

9.3.1 JMeter 介绍

JMeter 是一个用 Java 开发的跨平台开源软件，在 Windows 和 Linux 系统下均可以运行。可以从 Apache 官网下载 JMeter 最新版本，下载网址是：http://jmeter.apache.org/download_jmeter.cgi。

JMeter 安装比较简单，配置好 JDK，然后进入 JMeter 的 bin 目录，执行 jmeter.bat(Windows) 或者执行 jmeter.sh(Linux)，如果出现如图 9-32 所示的界面，刚表示安装成功。

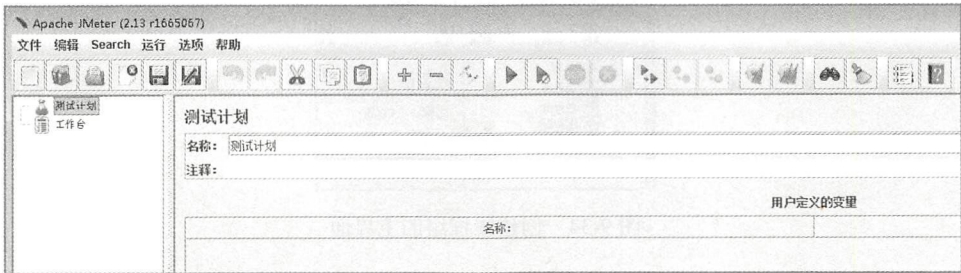


图 9-32 成功执行后显示的 JMeter 主界面

9.3.2 JMeter 脚本与优化

JMeter 和 Load Runner 一样也是基于协议的一款性能测试工具，在准备接口性能测试的时候，JMeter 更加容易使用。JMeter 从版本 2.9 起就废弃了 Web Services(SOAP)Request，而是使用 SOAP/XML-RPC Request。

1. 创建线程组

在添加 JMeter 脚本之前，首先需要添加线程组。线程组是用来创建 Vuser 的利器。线程组相当于虚拟用户组，一个线程组就相当于一个用户组，添加多个线程组就相当于模拟多个虚拟用户组对服务器进行压力测试。线程数就相当于虚拟用户数。

打开的 JMeter GUI 界面，用鼠标右键单击“测试计划”选项，在弹出的快捷菜单中选择“添加”→“Threads(Users)”→“线程组”选项，如图 9-33 所示。

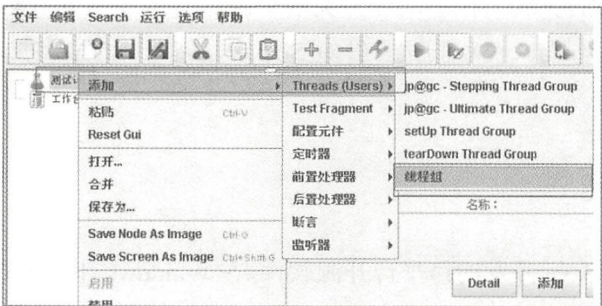


图 9-33 选择命令创建线程组

线程组创建完成后，主界面显示如图 9-34 所示。



图 9-34 创建线程组的主界面

选中一个线程组，在主界面的右侧区域可以看到线程组的配置页面，可以设置需要加压的频率和负载、调度器等，如图 9-35 所示。

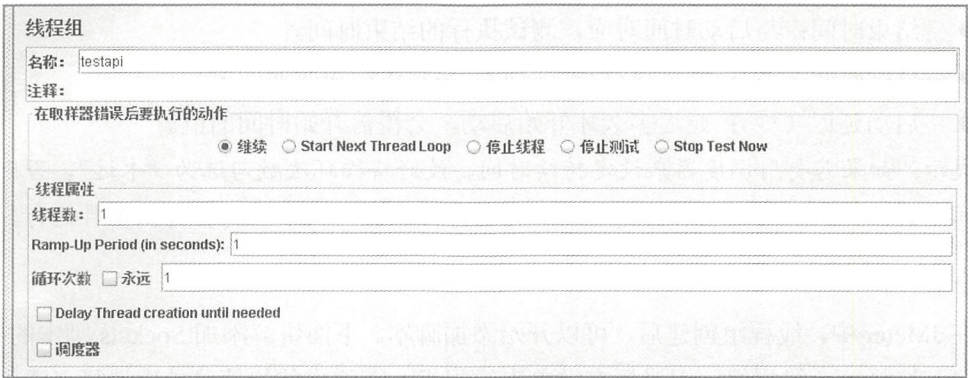


图 9-35 线程组配置

下面简单介绍一下各属性的作用和设置方式。

(1) 名称：线程组的名称，可以自定义修改，不过建议最好和项目名称一致，如“testapi”。

(2) 线程数：性能测试并发用户的数量，类似于 LoadRunner 中的 Vuser。

(3) Ramp-Up Period(in seconds)：设置 JMeter 多久建立起所有线程，默认值是 0。类似于 Load Runner 中的每秒 Vuser 用户加载个数。假设将 Ramp-Up Period 设置为 T 秒，全部线程数设置成 N 个，则 JMeter 将每隔 T/N 秒启动一个线程。

(4) 循环次数：脚本迭代次数。输入的数值数代表了要循环的次数。若勾选“永远”复选框，则脚本一直运行，直至手动停止脚本。

(5) Delay Thread creation until needed：延迟线程的创建直到需要时而不是在测试启动时创建。即之后的任何线程组延迟和加速时间为线程本身。这样可以支持更多的短生命期的线程，但不会有太多线程同时处于活动状态。

(6) 调度器：勾选后可以设置脚本的启动时间和结束时间。如果需要在将来某个时间点启动脚本，可以用这个选项，如图 9-36 所示。

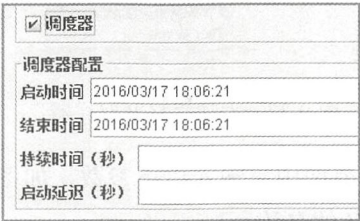


图 9-36 调度器设置

- 启动时间：测试执行的开始时间。

- 结束时间：与启动时间对应，测试执行的结束时间。
- 持续时间（秒）：测试执行持续时间，会覆盖结束时间的值。
- 启动延迟（秒）：延迟多久才开始启动，会覆盖开始时间的值。

提示：如果需要用调度器来设定持续时间，最好将循环次数勾选为“永远”，否则执行的时间是由循环次数的设置来决定的，系统运行 N 次循环之后，将会停止不再运行。

2. Sockets 协议脚本

在 JMeter 中，线程组创建后，可以开始添加脚本。下面讲解添加 Sockets 脚本的步骤。

(1) 选择一个线程组，用鼠标右键单击线程组，在弹出的快捷菜单中选择“添加”→“Sampler”→“TCP 取样器”选项，如图 9-37 所示。

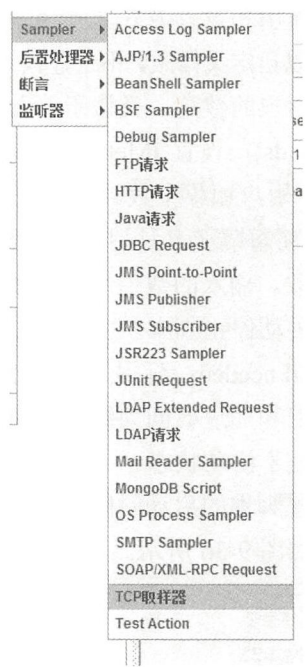


图 9-37 添加“TCP 取样器”

(2) 在打开的 TCP 取样器界面中，输入以下参数，如图 9-38 所示。

- 名称：输入被测试业务的名称。
- 服务器名称或 IP：输入被测系统 IP 或者域名。
- 要发送文本：输入 Sockets 请求的内容。

TCP取样器

名称: queryUserInfo_interface
注释: query接口

TCPClient classname:

Target Server
服务器名称或IP: 192.168.65.9 端口号: 9080
Connect: Response:

Re-use connection ☒ 设置无延迟 ☐

{1111111111111111}

要发送的文本

登陆配置
用户名:
密码:

图 9-38 TCP 取样器内容

3. HTTP 协议脚本

对于 HTTP 请求的接口,可以使用 badboy 进行脚本录制,然后将生成的脚本导入 JMeter 进行下一步调试,如图 9-39 所示。

HTTP请求

名称: http://192.168.51.107/mock/internal/mtp/processCnp.htm
注释:

Web服务器
服务器名称或IP: 192.168.51.107 端口号: 18080
Connect: Response:

Timeouts (milliseconds)
Connect: Response:

HTTP请求
Implementation: Java 协议: http 方法: POST Content encoding:

路径: /mock/internal/mtp/processCnp.htm

☒ 自动重定向 ☐ 跟随重定向 ☒ Use KeepAlive ☐ Use multipart form data for POST ☐ Browser-compatible headers

Parameters Post Body

名称	值	编码?	包含等于?
postUrl	http://192.168.65.8:8062/mas/internal/VVsMtpService	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
merchantId	104110045110028	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
merchantMemberCode	10011537062	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
terminalId	\$_Random(900000001,90040000,)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
bnType	PUR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Detail 添加 Add from Clipboard 删除 Up Down

同请求一起发送文件:

文件名称: 参数名称: MIME类型:

添加 浏览... 删除

Proxy Server
服务器名称或IP: 端口号: 用户名: 密码:

其他任务
☐ 从HTML文件获取所有内含的资源 ☐ Use concurrent pool, Size: 4 ☐ 用作监视器 ☐ Save response as MD5 hash?

Embedded URLs must match: Source IP address:

图 9-39 对 HTTP 请求进行调试

4. Java 协议脚本

(1) 编写 JMeter Java 脚本，首先需要使用 Java 开发工具，如 Eclipse 创建 Java 类。在 Eclipse 工具中，创建一个 Java 工程，然后引入 jmeter 的两个 Jar 包。这两个 jar 包放置于 JMeter 的安装目录 lib/ext 下：

- ApacheJMeter_java.jar
- ApacheJMeter_core.jar

如果待测试类已经有生成的 jar 包，可以引入至 Java 工程的 build 路径内。如果有源代码，直接导入源码至 Java 工程内即可。

(2) 新建一个测试类，它需要继承 AbstractJavaSamplerClient 抽象类并重写其抽象方法。

- public Arguments getDefaultParameters(): 设置可用参数及默认值。
- public void setupTest(JavaSamplerContext arg0): 每个线程测试前执行一次，做一些初始化工作。
- public SampleResult runTest(JavaSamplerContext arg0): 测试的实际执行函数。
- public void teardownTest(JavaSamplerContext arg0): 测试结束时调用。

测试类执行的先后顺序为：getDefaultParameters() → setupTest(JavaSamplerContext context) → runTest(JavaSamplerContext context) → teardownTest(JavaSamplerContext context)。

测试类常用的方法有以下几种：

- addArgument("name","value"): 定义参数并给参数赋值。
- sampleStart(): 定义事务的开始，类似于 LoadRunner 的 lr_start_transaction。
- sampleEnd(): 定义事务的结束，类似于 LoadRunner 的 lr_end_transaction。
- setSuccessful(true、false): 设置运行结果的成功或失败，JMeter 统计成功失败的次数，在聚合报告中能够体现。类似于 LoadRunner 的 lr_end_transaction 中的 LR_PASS 和 LR_FAIL。

测试类完成后把类打包，导出为可执行 jar 文件，并将 “*.jar” 文件复制到 JMeter 的安装目录 lib/ext 下。

(3) 运行 JMeter，添加一个线程组，然后在该线程组下面添加一个 Java 请求 Sampler，在 Java 请求的类名称中选择刚创建的类。

(4) 通过上面的例子可以发现，使用 JMeter 自定义 Java 测试代码，配合 JMeter 自带的函数，就可以实现 LoadRunner 中 “Java Vuser” 协议的绝大多数功能，而且是没有用户

数限制并完全免费的。

5. 脚本参数化

对于有些 JMeter 脚本，部分字段需要参数化，参数化时可以使用 CSV 文件配置。选中一个请求，右键单击 HTTP 请求，在弹出的快捷菜单中选择“添加”→“配置元件”→“CSV Data Set Config”选项，如图 9-40 所示。

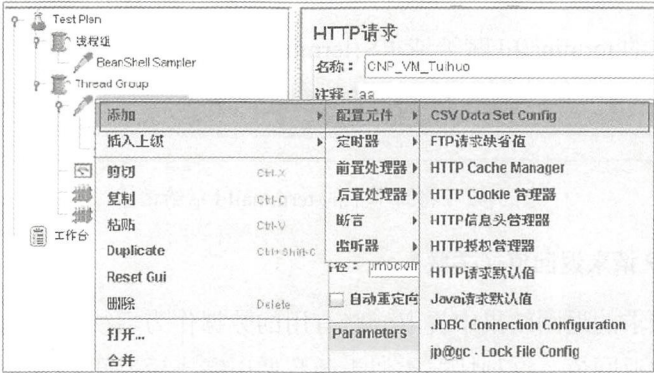


图 9-40 CSV Data Set Config 选项

(1) 对于新增加的 CSV Data Set Config，需要配置以下 3 个参数，如图 9-41 所示。

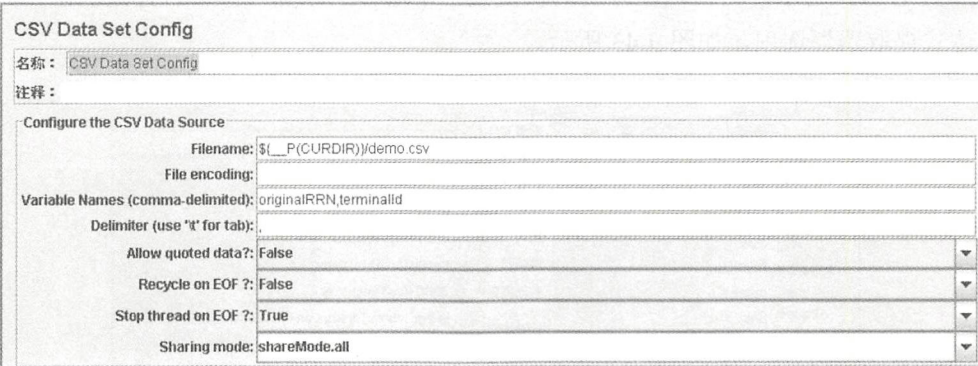


图 9-41 CSV Data Set Config 配置

- FileName: 输入参数化时使用的 CSV 文件的名称 demo.csv。
- Variable Names(comma-delimited): demo.csv 中有多个参数列表时是使用逗号分隔的。

- Delimiter(use ‘\t‘ for tab): 设置 demo.csv 文件中的数据作为分隔符。

根据 VariableNames(comma-delimited)项所填，从左往右为：originalRRN、terminalId。各参数中间用逗号隔开，参数可以从数据库查询。demo.csv 不需要像 LoadRunner 那样添加参数名，仅用参数值即可，示例如下：

```
ptsender_1,abc.1234
ptsender_2,abc.1234
```

(2) 将脚本中的 terminalId 赋给请求\${terminalId}，如图 9-42 所示。

名称:	值	编码?	包含等于?
terminalId	\${terminalId}	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
bnType	RFD	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

图 9-42 将脚本中的 terminalId 赋给请求

6. 提取 HTTP 请求返回值（关联）

有时需要从请求的返回结果中提取一些有用的数据作为后续脚本的输入参数，这时就需要提取请求中的返回值。这种情况类似于函数调用完毕后返回一个值作为其他函数的输入参数，也相当于 LoadRunner 中的关联。

(1) 在 HTTP 请求后添加一个正则表达式提取器，名称为“test 提取器”。在主界面中，选中一个 HTTP 请求，右击，在弹出的快捷菜单中选择“添加”→“后置处理器”→“正则表达式提取器”选项，如图 9-43 所示。

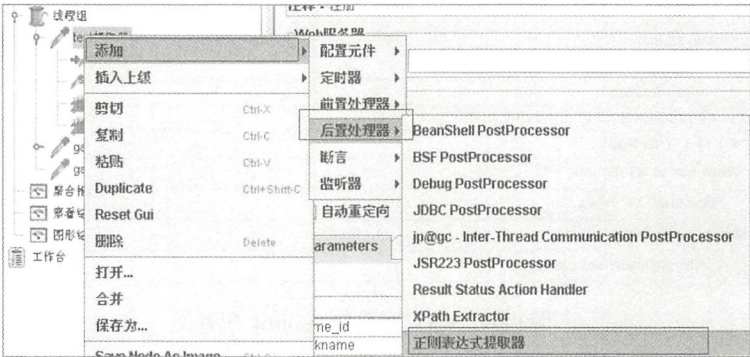


图 9-43 添加正则表达式提取器

(2) 打开正则表达式提取器界面，在其中设置以下参数，如图 9-44 所示。

- 引用名称: HTTP 请求从服务器返回的报文中将捕获到的值赋给 user_token 参数，

此参数会在后面的请求中使用。

- 正则表达式：提取左边界 “`"user_token":`” 和右边界 “`"`” 之间的内容，圆括号() 表示提取的内容。
- 模板：模板用`$$`引用起来，如果在正则表达式中有多个提取表达式（多个括号括起来的内容），则可以用`1`、`2`等，表示解析到的第几个值。正则表达式的提取模式序号从 1 开始，值 0 对应的是整个匹配的表达式。示例中的表达式使用`1` 模板，表示第一个圆括号内要取的值。
- 匹配数字：0 代表随机，-1 代表所有，其余正整数代表第几个匹配的内容，该数值是第一次出现并且唯一。
- 默认值：正则匹配失败时，设置的默认值。

正则表达式提取器

名称：

正则表达式提取器

注释：

get user_token

Apply to:

☐ Main sample only

☐ Sub-samples only

☒ Main sample and sub-samples

☐ JMeter Variable

要检查的响应字段

☒ 主体

☐ Body (unescaped)

☐ 信息头

☐ URL

☐ 响应代码

☐ 响应信息

引用名称：

user_token

正则表达式：

"user_token": "(+?)"

模板：

\$1\$

匹配数字（0代表随机）：

缺省值：

图 9-44 设置正则表达式提取器的参数

7. 设置断言

对于一个可以进行测试的接口测试脚本需要考虑的是：脚本运行后如何知道在高压力情况下被测业务有没有出错呢？这时就需要对返回的结果进行判断。JMeter 判断测试结果的方法是采用响应断言，类似于 LoadRunner 中的检查点。添加响应断言的操作步骤如下。

（1）在 HTTP 请求后插入一个响应断言，用于判断返回结果是否正确。在主界面中，选择一个 HTTP 请求，使用鼠标右键单击，在弹出的快捷菜单中选择“断言”→“响应断言”选项，打开响应断言界面如图 9-45 所示。

（2）设置完参数后，单击“添加”按钮，在“要测试的模式”下方的文本框中输入断言字符串，如“交易成功”，作为验证 HTTP 成功的指标，如图 9-46 所示。

• 351 •

仅供非商业用途或交流学习使用

响应断言

名称：

响应断言

注释：

Apply to:

☒ Main sample only

☐ Sub-samples only

☐ Main sample and sub-samples

☐ JMeter Variable

要测试的响应字段

☒ 响应文本

☐ URL样本

☐ 响应代码

☐ 响应信息

☐ Response Headers

☐ Ignore Status

模式匹配规则

☒ 包括

☐ 匹配

☐ Equals

☐ Substring

☐ 否

要测试的模式

要测试的模式

添加

删除

图 9-45 响应断言界面

响应断言

名称：

响应断言

注释：

Apply to:

☒ Main sample only

☐ Sub-samples only

☐ Main sample and sub-samples

☐ JMeter Variable

要测试的响应字段

☒ 响应文本

☐ URL样本

☐ 响应代码

☐ 响应信息

☐ Response Headers

☐ Ignore Status

模式匹配规则

☒ 包括

☐ 匹配

☐ Equals

☐ Substring

☐ 否

要测试的模式

要测试的模式

交易成功

添加

删除

图 9-46 响应断言

• 352 •

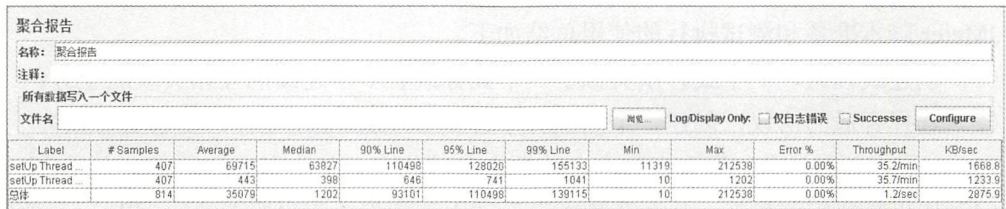
仅供非商业用途或交流学习使用

9.3.3 收集监控数据

JMeter 也可以和 Load Runner 一样收集测试中脚本的响应时间、TPS、吞吐量和失败率。接下来讲解如何添加监控工具。在 JMeter 中添加监控视图比较简单，能监控的性能指标也比较完善，所有的监控器都位于“添加”→“监听器”选项里。

如果要查看响应时间、TPS、吞吐量和失败率，那么可以添加一个 Aggregate Report（聚合报告），具体步骤如下：

- （1）选择一个线程组，在右键快捷菜单中单击“添加”→“监听器”→“聚合报告”。
- （2）测试执行完毕后，在聚合报告中可以看到响应时间、TPS 和吞吐量等信息，如图 9-47 所示。



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	kB/sec
setUp Thread	407	69715	63827	110498	128020	155133	11319	212538	0.00%	35.2/min	1668.8
tearDown Thread	407	443	398	646	741	1041	10	1202	0.00%	35.7/min	1233.9
总计	814	35079	1202	93101	110498	139115	10	212538	0.00%	1.2/sec	2875.9

图 9-47 聚合报告

提示：其他 JMeter 插件也可以用来监控，如果需要的话可以去 JMeter 官网或一些开源论坛查找，也可以考虑写 JMeter 插件。比如 JMeterPlugins_patch.jar 和 Jvm_MQ_Collect_Plugin.jar。

9.3.4 运行测试


在线程组设置中，调整虚拟用户数，单击“启动”按钮就可以开始测试了，如图 9-48 所示。分析结果有硬件资源分析、JVM 分析、网络连接分析、数据库分析和中间件分析等。





图 9-48 运行测试

在 JMeter 主界面的工具栏中，可以单击“运行”按钮，执行测试，或者选择“运行”菜单项。执行脚本时，选择菜单项“运行”→“启动”，或者单击工具栏中的▶按钮。停止脚本时，选择菜单项“运行”→“停止”，或者单击工具栏中的●按钮。

在分布式性能测试中，需要在远程机器上部署和调用脚本，有以下两种方式：

- 启动一台远程机器：选择菜单项“运行”→“远程启动”→“远程机器的 IP”，被选中的负载机开始执行测试，可以逐个选择其他需要的负载机。
- 启动所有的负载机实施集群性能测试：选择菜单项“运行”→“远程全部启动”，或者单击工具栏中的图标，所有列表上的负载机都会执行测试脚本。

每次测试执行完毕后，需要清理测试执行数据。清除本次测试产生的缓存数据可以按 CTRL+D 快捷键，或单击工具栏上的图标，如果清除所有的缓存数据可以单击工具栏上的按钮。

9.3.5 JMeter 使用小结

JMeter 脚本准备和测试执行的使用总结如下。

- (1) 创建线程组，一个线程组类似于一个业务或者多个连续的互相关联的业务。
- (2) 选择合适的协议，创建一个测试脚本（和 LoadRunner 一样，要选对协议）。
- (3) 如果需要参数化，那么在脚本之前插入参数化.csv 文件（LoadRunner 为.dat 文件）。
- (4) 如果需要提取数据，那么在脚本之后插入正则表达式提取器（LoadRunner 使用关联函数）。
- (5) 添加一个响应断言（LoadRunner 使用函数来捕获返回值）。
- (6) 添加合适的监控器（LoadRunner 比较智能，只要开启性能测试场景，大部分性能参数自动获取）。
- (7) 执行测试脚本，获得测试结果（LoadRunner 有更直观详细的测试结果）。JMeter 的测试计划结构如图 9-49 所示。



图 9-49 测试计划

9.4 性能测试框架搭建

JMeter 在进行长时间的测试后，数据信息会越来越多，这些数据在测试完成之后进行读取和分析时很不直观。如果能有一个实时的图形结果来展示这些测试数据对测试人员来说将会更有利。使用 Backend Listener 插件，可以不用等待整个测试执行完成就能查看测试结果和数据，在整个测试过程中，数据可以实时输出并以动态图表的形式在前端展示，极大方便了性能测试人员对实时性能测试数据的监控，加上 Grafana 强大的图表展示功能，每个测试人员都可以搭建出自己喜欢的图形化页面，并实时与其他人员共享，完成对整个性能测试过程中数据的实时监控。如图 9-50 所示，就是这个性能测试框架的组成。

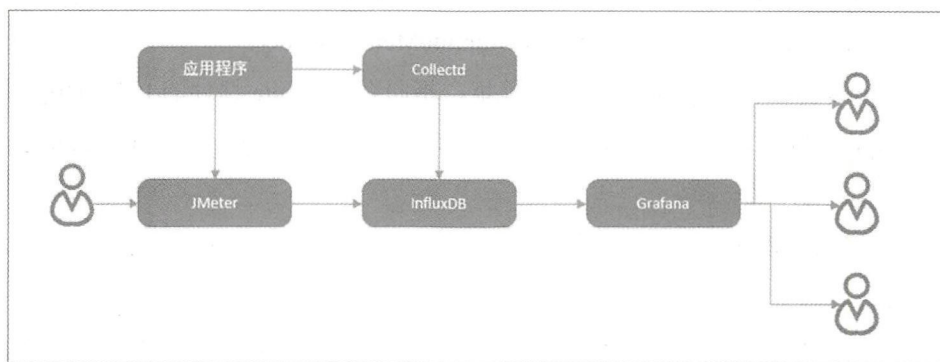


图 9-50 JMeter+Influx+Grafana 性能测试框架

提示：JMeter 的 2.1 以上版本支持监听器配置。

9.4.1 JMeter 配置监听器

JMeter 执行性能测试后产生的 Metrics 数据需要传送至 Graphite 的后端，可以通过添加一个 Backend Listener 监听器，使用监听器中的 GraphiteBackendListenerClient 类实现。Backend Listener 监听器把测试过程中的实时结果写到时序数据库（InfluxDB、Graphite 等）中，这里以 InfluxDB 为例来保存数据，如图 9-51 所示。

Backend Listener

名称: Backend Listener

注释:

Backend Listener implementationorg.apache.jmeter.visualizers.backend.graphite.GraphiteBackendListenerClient

Async Queue size5000

Parameters

名称:	值
graphiteMetricsSender	org.apache.jmeter.visualizers.backend.graphite.TextGraphiteMetricsSender
graphiteHost	10.108.12.132
graphitePort	2003
rootMetricsPrefix	jmeter.
summaryOnly	false
samplersList	SessionExamples,sessionExample3,index,sessionExample1,sessionExample4,sessionExample5,sessionExample2Post
percentiles	90;95;99

图 9-51 Backend Listener 监听器

Backend Listener 的参数配置如下。

- graphiteMetricsSender: Graphite 度量的发送类，有两种类供选择，即 org.apache.jmeter.visualizers.backend.graphite.TextGraphiteMetricsSender 和 org.apache.jmeter.visualizers.backend.graphite.PickleGraphiteMetricsSender。
- graphiteHost: Graphite 服务器的地址，指向有 Graphite 插件的 InfluxDB。
- graphitePort: Graphite 服务器的端口，和 InfluxDB 的 input_plugins.graphite 配置文件中的端口号一致。
- rootMetricsPrefix: 发送给后台的度量前缀，默认为“jmeter.”，此处的“.”是必需的。
- summaryOnly: 发送统计信息，默认为 true，仅统计数据。如果设置为 false，则可以发送详情。
- samplersList: 采样器列表，添加的采样器名字，多个采样器以分号隔开。
- percentiles 收集响应时间的百分比度量，以分号隔开。默认的百分比是“90;95;99”，即 90%、95%和 99%。

9.4.2 InfluxDB 数据库配置

在 InfluxDB 管理站点，创建数据库“jmeter”，如图 9-52 所示。

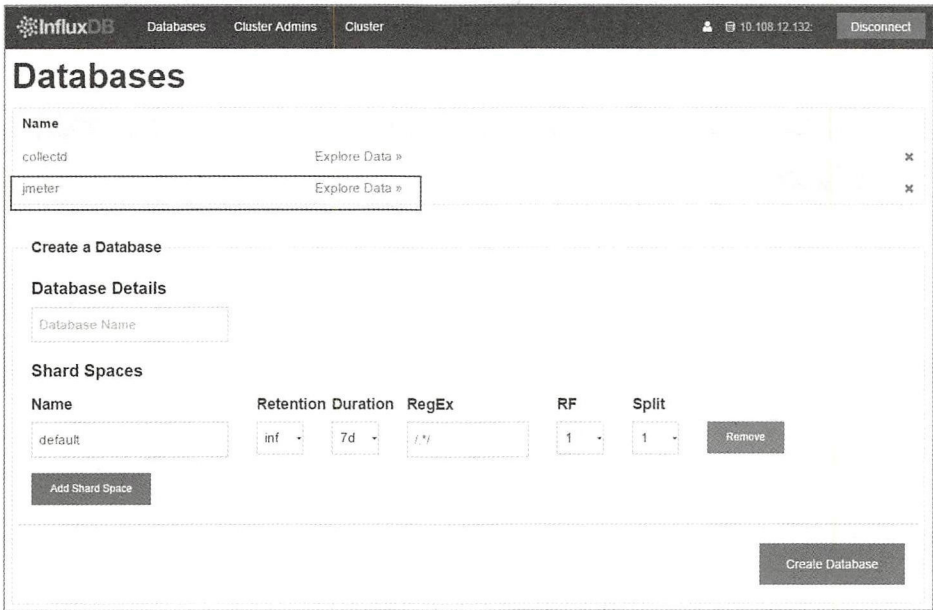


图 9-52 创建数据库 “jmeter”

9.4.3 InfluxDB Graphite Listener 配置

在 InfluxDB 配置文件中，启用 Graphite Listener，编辑文件/opt/influxdb/shared/config.toml 或者/usr/local/etc/influxdb.conf，找到 “input_plugins.graphite”，代码设置如下，其中的 port 和 address 对应于 Backend Listener 的参数 graphitePort:

```
# Configure the graphite api
[input_plugins.graphite]
enabled = true
#address = "0.0.0.0" # If not set, is actually set to bind-address.
port = 2003
database = "jmeter" # store graphite data in this database
# udp_enabled = true # enable udp interface on the same port as the tcp interface
```

9.4.4 查看 InfluxDB 结果

查询 “jmeter.all.a.count”，可以查看服务器每秒处理的请求数，如图 9-53 所示。

质量全面管控：从项目管理到容灾测试

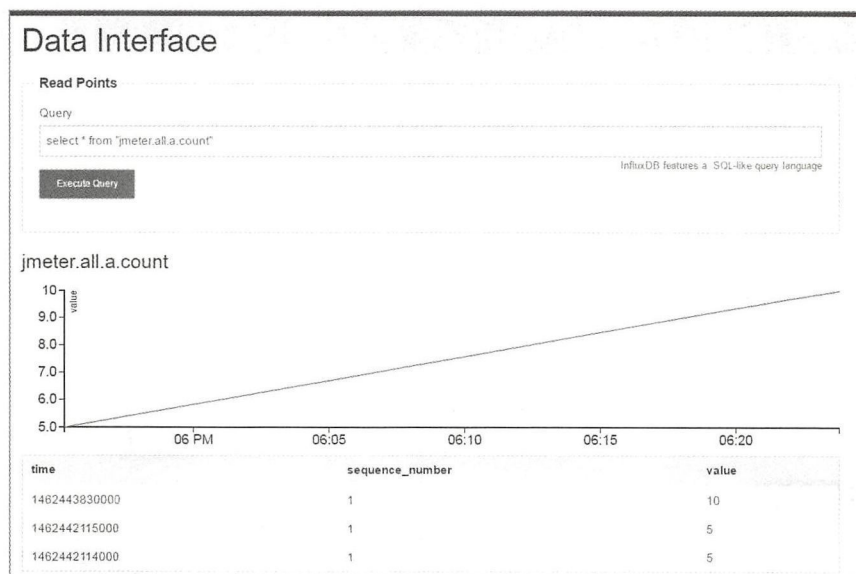


图 9-53 查看 InfluxDB 结果

9.4.5 Grafana 配置

在 Grafana 中，新增数据源，在“Database”文本框中输入 9.4.2 节中创建的数据库“jmeter”，如图 9-54 所示。单击“Test Connection”按钮，确保已经连接到 InfluxDB。

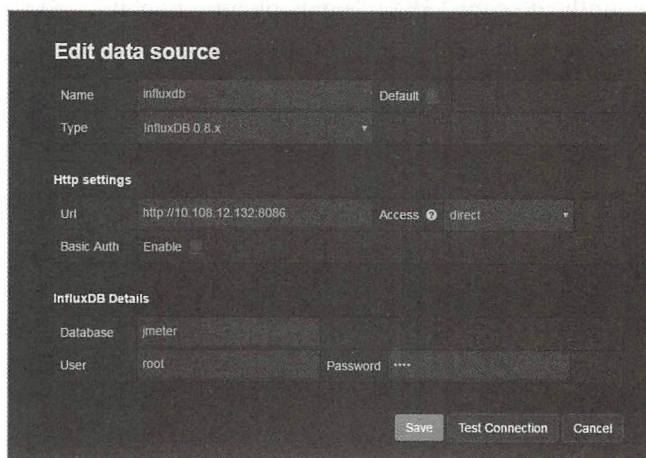


图 9-54 新增并设置数据源

至此，JMeter 数据导入 InfluxDB，通过 Grafana 前端展示的整个过程已配置成功，接下来就可以在 Grafana 上自定义图表，如图 9-55 所示。具体的图形配置和展示方法，可以参考 Grafana 的官方文档。

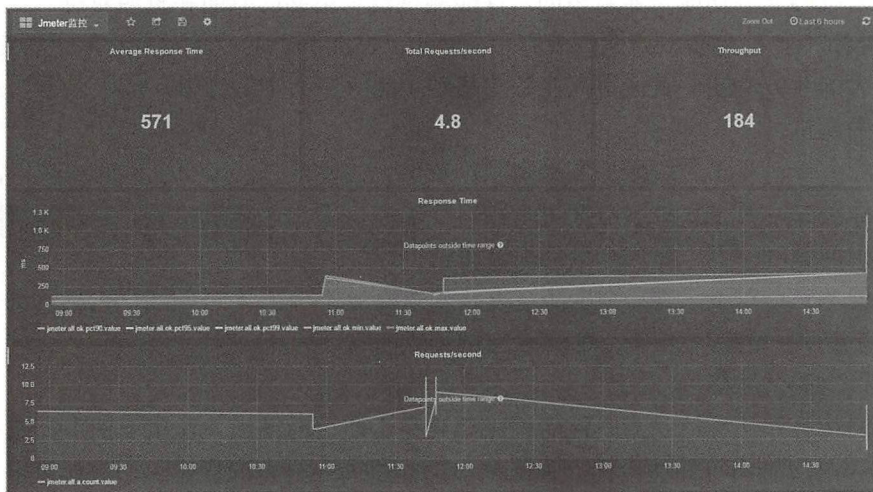


图 9-55 Grafana 展示 JMeter 数据

9.5 性能测试实战

本节以实例项目“账户余额变更项目”来介绍性能测试的步骤，此项目是一个账户余额变更系统，账户有多个级别，包括一级账户、二级账户、三级账户和四级账户。其中各级别账户和检查有关联。

- 一级账户：有多个二级账户，变更余额时不需要对账户进行额度检查。
- 二级账户：有多个三级账户，变更余额时需要进行 2 个额度检查，包括一级和二级账户的额度检查。
- 三级账户：有多个四级账户，变更余额时需要进行 3 个额度检查，包括一级、二级和三级账户的额度检查。

9.5.1 明确测试需求

研发人员或项目经理填写性能测试需求申请表，填写完并提交给性能测试人员，测试

质量全面管控：从项目管理到容灾测试

人员进行需求申请表评审，评审通过开始编写性能测试方案。测试方案制定时，需要设定测试范围和目的，收集测试环境信息。

（1）确定测试的业务范围

一个系统的业务功能是非常多的，性能测试没有足够的时间和资源对每一个功能点都进行压力测试，如何确定测试范围变得至关重要。性能测试的业务需要选取最关键的业务，如并发最大的业务功能、其他重点业务的基础功能等。在本实例中，选取以下 3 个关键业务进行压力测试。

- 业务一：对于一级账户变更余额，要求无额度检查，账户余额的增减正常。
- 业务二：对于二级账户变更余额，要求不少于 2 个额度检查，包括一级和二级账户的额度检查，账户余额的增减正常。
- 业务三：对于三级账户变更，交易检查额度不少于 4 个额度，包括一级、二级和三级账户的额度检查，账户余额的增减正常。

（2）设定测试目标

测试的指标和指标值是判断性能测试结果是否通过的准绳。在本实例中，获得测试接口的 TPS、响应时间等性能指标，找出接口性能瓶颈并给出调优建议。需要设定的性能测试通过准则如表 9-2 所示。

表 9-2 性能测试通过准则

序号	指标类别	指标	指标值
1	系统处理能力（TPS，响应时间）	业务一“一级账户变更余额”的 TPS 和响应时间	100 笔/s ≤0.1s
2		业务二“二级账户变更余额”的 TPS 和响应时间	50 笔/s ≤0.2s
3		业务三“三级账户变更余额”的 TPS 和响应时间	30 笔/s ≤0.3s
4	服务器资源利用率	CPU 占用率	≤80%
5		内存使用率	≤80%
6		I/O 使用率	≤80%
7	系统稳定性	系统无故障运行时间	48 小时

（3）明确环境资源配置

设置好测试范围和目标后，需要收集生产环境和实际测试环境之间的标准差异，最好是 1:1 的比例。如果达不到生产环境标准，需要计算差异系数来预估结果。本实例的环境资源配置如表 9-3 所示，包括待测服务器的操作系统、CPU、内存和硬盘的信息。

表 9-3 环境资源配置

序号	机器类型	用途	操作系统	CPU 处理器	物理内存	硬盘
1	虚拟机	Tomcat	RHEL5.6	双核 Intel (R) Xeon (R) CPU E5-2603 处理器 (1.80GHz)	16GB	300GB
2	物理机	MQ	RHEL5.6	八核 Intel(R) Xeon(R) CPU E5506 处理器 (2.13GHz)	16GB	300GB
3	物理机	Oracle	RHEL5.6	32 核 Intel (R) Xeon (R) CPU E7-4830 处理器 (2.13GHz)	126GB	1.8TB

9.5.2 选取测试方法和策略

(1) 收集完测试需求，开始制定测试方案。

本实例有三个级别的账户，每一级账户的接口测试需要进行单场景测试来找出最佳用户数，然后整合三个级别账户的接口，执行混合场景测试。测试方法和策略在性能测试方案中体现，如表 9-4 所示。

表 9-4 选取测试方法和策略

场景编号	接口名称	场景类型	并发用户数	执行时间	用户加载方式	集合点设置	执行策略	备注
DEMO-01	(一级账户体系) 余额变更	单场景	单个用户	10 分钟	同时加载	无	探索测试	
DEMO-02	(一级账户体系) 余额变更	单场景	1000 个用户	10 分钟	每 15 秒加载一个用户	无	压力测试	找出最大用户数，并发用户数视系统能力而定
DEMO-03	(一级账户体系) 余额变更	单场景	最佳用户数	1 小时	同时加载	无	负载测试	
DEMO-04	(一级账户体系) 余额变更	单场景	最佳用户数	10 分钟	同时加载	有	集合点测试	
DEMO-05	(一、二、三级账户体系) 余额变更	混合场景	最佳用户数	1 小时	同时加载	无	负载测试	

质量全面管控：从项目管理到容灾测试

（2）准备测试数据

根据性能需求预估测试的数据量，利用数据库的存储过程 Store Procedure 产生大量的测试数据，注入到性能测试环境的数据库中。数据库导入的账户和余额数据量罗列如下。

- 一级账户数据：10000 个主账户、10000 个余额和 10000 个余额属性。
- 二级账户数据：1000 个主账户，2000 个汇总账户、2000 个附属账户和 10000 个余额检查。
- 三级账户数据：1000 个主账户、3000 个汇总账户、3000 个附属账户和 28000 个余额检查。

（3）选取压力测试和监控工具

此项目性能测试的目标是监控接口的 TPS、响应时间等数据，还要监控数据库和硬件资源如 CPU、内存、网络指标等。本实例中选用以下几个工具。

- HP LoadRunner：性能测试工具，对系统增加负载，采集 TPS 和响应时间等指标。
- Toad 和 Oracle EM 企业管理器：获取 Oracle 数据库的 AWR 报告，监控数据库 CPU、负载、索引、Top X 使用频率的 SQL 语句、Top X 的会话等。
- Visual VM：监控 DB 的连接数、JVM 堆内存使用情况和线程情况。
- nmon：监控服务器的硬件资源，如 CPU、Memory、Disk 和网络等。

9.5.3 准备测试脚本

选择 LoadRunner 后，就要准备性能测试的脚本，脚本使用 Virtual User Generator 工具创建并优化。本实例涉及 3 个测试接口，故可以设计 3 个测试用例，也就是 3 个测试脚本。测试脚本比较简单，是 WebServices 协议的脚本，本章前面介绍的步骤可以快速完成。

9.5.4 执行与分析测试结果

在 LoadRunner Controller 中，按照 9.5.2 小节中的场景需求，添加 LoadRunner 测试脚本，设置场景，包括每个脚本对应的并发用户数、Vuser 启动方式、测试持续时间、用户加载和减少的速度等。然后执行和监控测试过程，最后产生测试结果，并对测试结果进行以下分析。

（1）如图 9-56 所示为一级账户在变更余额时的响应时间图，上方第二条曲线为提交余

额变更的事务响应时间。当并发用户达到 500 个时，响应时间大幅度增加，但还没有超过 2 秒。当继续增加并发用户人数到 600 个左右时，响应时间已经超出 3 秒，超出预设的响应时间指标，表明系统性能未达标，需要进行优化。

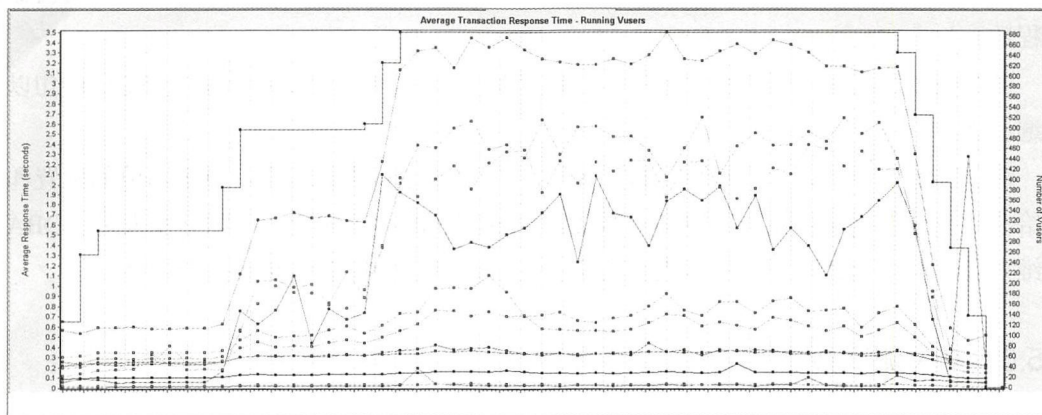


图 9-56 响应时间测试结果

(2) 在 Oracle 数据库的 AWR 报告中，发现一级账户在做余额更新操作时查询账户余额的 SQL 语句执行时间需要 3~4 秒，已经超过性能指标阈值，如图 9-57 所示，需要进一步优化 SQL 语句，检查数据库的表结构和索引。

SQL ordered by Elapsed Time (Global)																
<ul style="list-style-type: none"> Captured SQL account for 87.6% of Total DB Time (s): 25.279 Captured PL/SQL account for 0.1% of Total DB Time (s): 25.279 																
SQL Id	Total							Per Execution								
	Elapsed (s)	CPU (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	Execs	Elapsed (s)	CPU (s)	IOWait (s)	Gets	Reads	Rows	Cluster (s)	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	3.774.38	1.775.67	2.007.16	221,632,316	221,604,432	6,648	0.00	6,647	0.57	0.27	0.30	33,343.21	33,339.01	1.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	2.880.11	1.811.23	1.074.05	221,656,072	117,237,768	6,648	0.00	6,648	0.43	0.27	0.16	33,341.77	17,635.04	1.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	2.813.06	1.734.87	1.080.19	211,106,972	111,717,693	6,647	0.00	6,648	0.42	0.26	0.16	31,754.96	16,804.71	1.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	2.672.14	1.602.29	1.073.11	211,042,178	111,656,725	6,647	0.00	6,648	0.40	0.24	0.16	31,745.21	16,795.54	1.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	2.492.65	1.489.41	1.005.84	192,657,733	101,972,748	6,646	0.00	6,648	0.37	0.22	0.15	28,979.80	15,338.86	1.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	2.423.23	1.440.17	985.60	192,656,937	101,917,675	6,353	0.11	6,646	0.36	0.22	0.15	28,988.40	15,335.19	0.96	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	2.354.63	1.809.41	166.05	196,818,549	41,604	6,008	475.50	6,007	0.39	0.30	0.03	32,764.87	6,993	1.00	0.08	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	1.195.10	1.191.73	0.00	8,620,649	0	349	0.00	349	3.42	3.41	0.00	24,701.00	0.00	1.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	1.116.00	1.091.14	0.17	196,817,284	42	180,240	20.51	6,908	0.19	0.18	0.00	32,759.20	0.01	30.00	0.00	
select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)	112.15	68.29	43.93	8,494,906	4,521,983	0	0.00	293	0.38	0.23	0.15	28,992.85	15,433.39	0.00	0.00	

图 9-57 AWR 报告

从 AWR 报告中，可以观察到 Per Execution 区域下的 Elapsed(s)显示了 SQL 语句的执行时间为 3.42 秒，SQL Text 列显示的 SQL 语句为：select count(*) from(select * from VIEW_ACCOUNT_BAL a where acct_type = 1)。

质量全面管控：从项目管理到容灾测试

9.5.5 提出调优建议

针对以上问题，向开发人员提出以下调优建议：

（1）综合查看应用服务器的资源和线程数，建议增加应用服务器，提高负载均衡器的线程数。

（2）数据库发生了两次全表遍历，建议增加 SQL 缓存，修改有问题的 SQL 语句以减少遍历次数。

这里需要注意的是，在调优阶段，测试人员起辅助和协调作用，对于当前的配置和状态给予建设性的意见。具体的调优参数还需要各方面的专家配合，比如开发人员、DBA、运维人员等。在调优期间，测试人员要进行回归测试以验证调优参数是否合格。

9.5.6 交付测试报告

性能测试结束后，需要提交以下 3 个报告。

（1）账户余额变更项目性能测试方案与计划：对项目进行评估，设定测试范围和目标，选择测试方案，制定测试计划，与项目干系人达成共识。

（2）账户余额变更项目性能测试日报：每日提交测试报告，汇报整个性能测试的进度状态，是否遇到障碍，如环境配置、人员安排、系统不稳定等问题。

（3）账户余额变更项目性能测试报告：在每一轮性能测试结束后交付一份测试报告，汇报整个测试的过程，分析实际测试结果，提出调优建议。

9.6 性能调优

性能测试结果分析之后，召集开发人员、DBA 和运维人员，定位瓶颈并进行调优。本节对性能调优进行简单介绍，使读者了解一些常见的性能瓶颈和调优方案。

9.6.1 CPU 使用率过高

场景：如图 9-58 所示的最上方的曲线是在某一次压力测试中应用服务器的 CPU 使用率。当并发用户达到 100 个以上时，应用服务器的 CPU 使用率持续较高，平均值超过 75%。

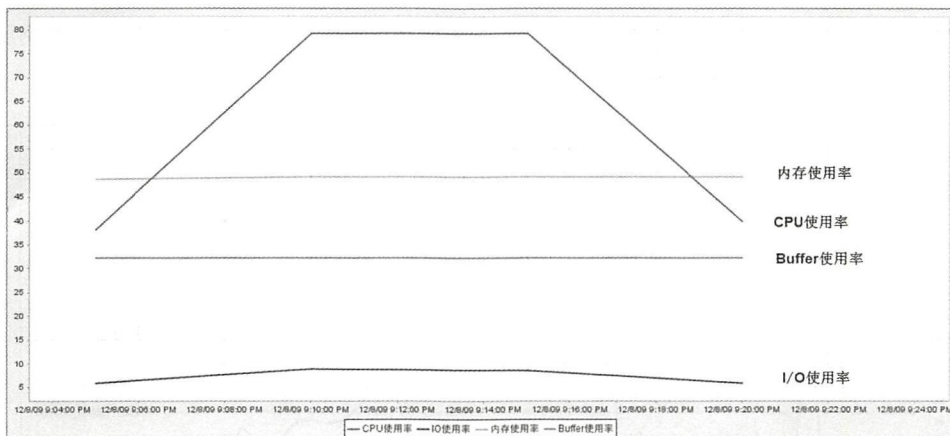


图 9-58 调优前 CPU%过高

优化建议：本次应用服务器的 CPU 处理器为 4 核处理器，频率为 $4 \times 1.6\text{GHz}$ 。如果要提高系统性能，主要方法有增加 CPU 数量、提高 CPU 主频或者组建系统集群来分担压力；也可以考虑修改应用程序的代码，提高应用程序本身的效率，减少单位进程的 CPU 资源的消耗。优化过后的 CPU 使用率比较稳定，几乎没怎么变动，如图 9-59 所示。

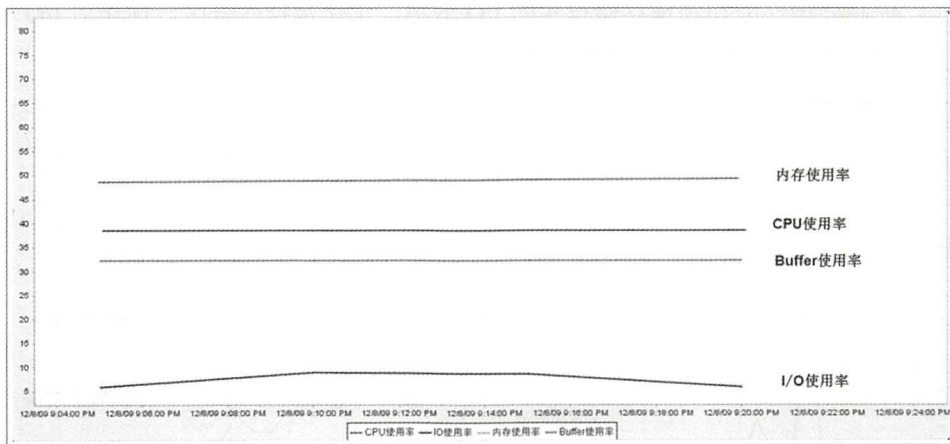


图 9-59 调优后 CPU 使用率正常

9.6.2 I/O 使用率过高

场景：在稳定性测试过程中，发现大量的并发用户执行的事务都失败了，初步诊断后

质量全面管控：从项目管理到容灾测试

发现数据库系统的 I/O 使用率持续为 100%，如图 9-60 所示，导致不能及时得到响应，已经超时 30 秒以上，收不到报文的返回。

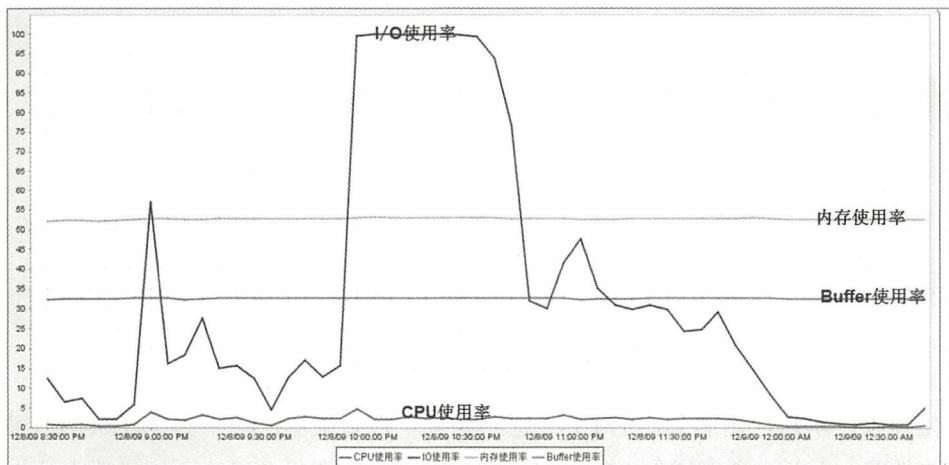


图 9-60 I/O 使用率过高

优化建议：使用 `ls -l` 列出来的文件可以发现哪些库文件及日志文件发生大量的 I/O 操作，提高处理性能和物理存储设备的 I/O 效率，优化数据库分区。优化后 I/O 使用率有所改变，处于较低使用水平，如图 9-61 所示。

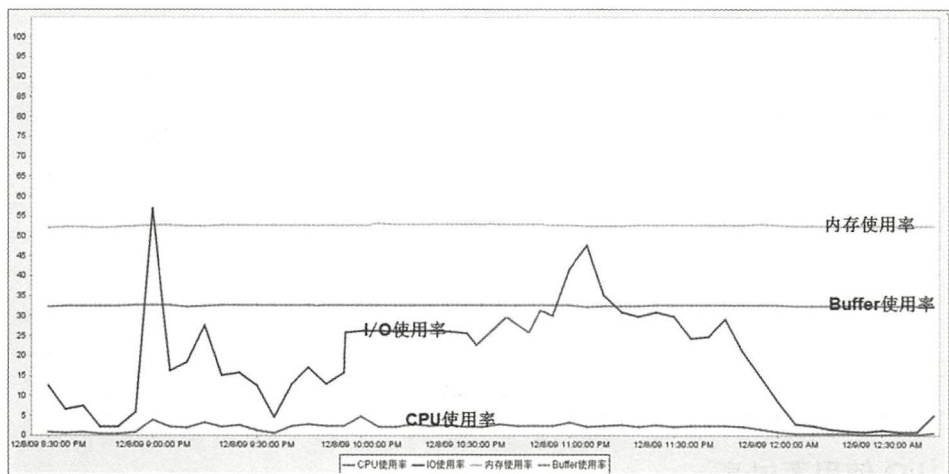


图 9-61 调优后 I/O 使用率正常

9.6.3 进程数调整

场景：根据测试环境的硬件配置情况，启动不同的进程个数，也会影响到系统性能。如图 9-62 所示，测试过程中并发用户数为 100，初始进程数为 2，此时 TPS 保持在 3 左右；当增加进程数至 8 个时，TPS 逐步提升，最终保持在 8 左右。

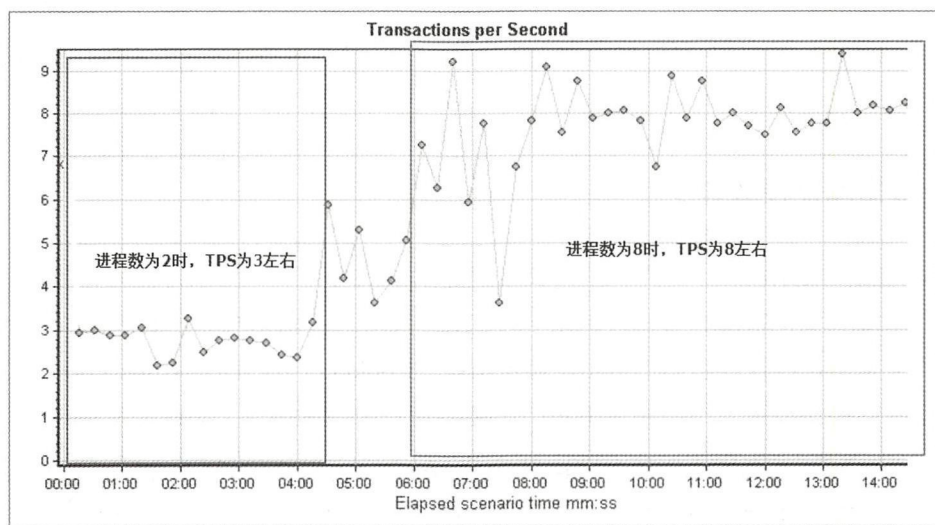


图 9-62 进程数调整前后的 TPS 趋势图

优化建议：进程数过少会导致拥堵，过多会消耗更多的系统资源，多次测试后找出最佳平衡点。在测试环境中，启动的进程数为 6~8 个比较合理，系统性能达到最大化。

9.6.4 线程不安全

场景：在压力测试过程中，发现某系统在多人操作时，债权匹配接口调用失败，返回 400 错误。以下是后台应用服务器日志的部分片断，打印了接口的调用参数。在日志中发现参数 loan_term 的值与输入的值不一致。原本输入的 loan_term 值为 18，但日志中打印的却是 24：

```
INFO:      债      权      匹      配      ,      调      用      参
数 :{"borrow_amount":"30000.00","card_id":"XXXXXXXXXXXXXXXXXXXX","client_id":2611112,"client_name"
:"      花      花      世      界
","client_phone":"13988888888","contract_amount":"85714.22","contract_numbe":"null","end_repay_date":"201
```



```
7-10-10","id":2801551,"loan_amount":60000.00,"loan_term
```

```
...  
INFO: 债权匹配, 转换结果:{"code":400,"msg":"未知错误, 联系管理员吧."}
```

优化建议：开发人员分析代码后，发现这个问题是由线程不安全造成的，在多个线程访问时，输入的参数值被覆盖了。为了解决问题，在程序的代码中增加了多线程的安全处理，避免并发取值冲突。另外，应用程序代码从数据库读取 `loan_term` 值来替代服务器读取错误值。

9.6.5 数据库连接数过少

场景：在单场景测试中，750 个并发用户登录系统时，TPS 达到 15，响应时间小于 2 秒，并且系统表现正常没有宕机。但是 1500 个并发用户登录系统时，TPS 为 29，但是系统异常响应延迟，后台日志如下所示，显示错误“Could not get JDBC Connection”。这表示当前数据库连接池已满，不能新建 JDBC 连接，需要增加连接池大小：

```
Caused by: org.springframework.jdbc.CannotGetJdbcConnectionException: Could not get JDBC  
Connection; nested exception is org.apache.commons.dbcp.SQLNestedException: Cannot get a connection, pool  
error Timeout waiting for idle object  
at  
org.springframework.jdbc.datasource.DataSourceUtils.getConnection(DataSourceUtils.java:80)  
at org.springframework.jdbc.core.JdbcTemplate.execute(JdbcTemplate.java:575)  
at org.springframework.jdbc.core.JdbcTemplate.query(JdbcTemplate.java:639)  
at org.springframework.jdbc.core.JdbcTemplate.query(JdbcTemplate.java:668)  
at org.springframework.jdbc.core.JdbcTemplate.query(JdbcTemplate.java:676)  
at org.springframework.jdbc.core.JdbcTemplate.queryForObject(JdbcTemplate.java:731)  
at org.springframework.jdbc.core.JdbcTemplate.queryForObject(JdbcTemplate.java:747)  
at org.springframework.jdbc.core.JdbcTemplate.queryForInt(JdbcTemplate.java:782)
```

调优方案：由于操作系统是 Oracle，可以使用以下 SQL 语句，查看数据库的当前连接数、最大连接数和并发连接数，发现当前数据库连接数等于最大连接数，另外数据库后台日志报错。

(1) 执行如下 SQL 语句，查看到数据库允许的最大连接数是 30：

```
select value from v$parameter where name = 'processes';
```

(2) 执行如下 SQL 语句，发现查看当前的数据连接数是 30。

```
select count(*) from v$sqlprocess;
```

于是将 DB 的最大连接数设置成 150，重新执行性能测试后，此错误消失。

9.6.6 数据导入慢

场景：某个批处理程序需要导入文本形式的账户信息到数据库中，其执行时间根据账户数据量成指数增长，如表 9-5 所示。

表 9-5 优化前的数据量与执行时间

数据量	执行时间
1 万	54 秒
10 万	30 分 35 秒

调优方案：采用以下几条方式进行调优后，执行时间有显著的提升，优化后的结果如表 9-6 所示。

- (1) 将批处理程序中的部分逻辑由复杂、费时的数据库存储过程转移到 Java 代码中，特别是用代码提前处理了有些账号信息。
- (2) 批处理程序与数据库环境分离，通过 SQL Loader 导入数据。
- (3) 将账户明细的全量数据改为增量数据，减少不必要的数据库导入。
- (4) 增大数据库的系统全局区（SGA），提高数据库的处理速度。

表 9-6 优化后的数据量与执行时间

数据量（万）	执行时间
40 万	16 分 18 秒
80 万	26 分 12 秒
200 万	75 分钟

9.7 要点回顾

本章要点回顾如下：

- 性能测试是模拟正常、峰值及异常负载条件来对系统的各项性能指标进行测试。
- 常见的性能测试包括负载测试、压力测试、容量测试和稳定性测试。
- 熟练掌握 Load Runner 和 JMeter 的使用。
- 讲述了 JMeter、InfluxDB 和 Grafan 框架，将 JMeter 的执行数据导入 InfluxDB，再通过 Grafana 展示出来，它解决了 Jmeter 在执行性能测试时的数据非实时问题。
- 性能测试实战的过程包括明确测试业务需求与目标、收集环境资源配置、选取测试方法与策略、准备测试数据和背景数据、选取性能测试和监控工具、准备测试脚本、执行和分析测试结果，提出调优建议和交付测试报告。

第 10 章

性能分析

在完成测试脚本、准备性能测试环境、功能测试验证通过后，就开始进入性能测试阶段。当测试场景运行时，需要收集业务性能数据（如业务响应时间和 TPS），监控服务器资源、数据库和网络等性能指标。

监控是性能分析的第一步，即通过观察各项性能指标的变化来判断性能问题。收集这些性能指标也是一个简单的分析过程，可以排除一些干扰因素，为后续进一步地性能分析做准备。

监控和分析的方法多种多样，这里只是抛砖引玉，图 10-1 所示是执行性能测试时监控和分析的全过程，步骤如下。

- （1）使用性能测试工具收集应用系统的业务性能数据，并监控数据有无性能异常。
- （2）观察服务器的硬件资源和性能状况，判断硬件是否存在瓶颈，监控数据包括 CPU、Memory、Disk、I/O、网络和数据库连接数等。
- （3）了解 JVM，分析 JVM 参数是否存在问题，监控 JVM 使用情况是否有异常。
- （4）深入分析问题根源，如分析 JVM、查询代码异常、使用 Oracle 分析工具分析数据库问题，结合第（1）步的业务性能，最后确定瓶颈所在的位置。
- （5）找出可优化的参数和硬件资源，调整之后，进行调优测试。
- （6）调优后，对先前的性能问题进行回归测试，监控性能数据，验证性能问题是否已经解决。

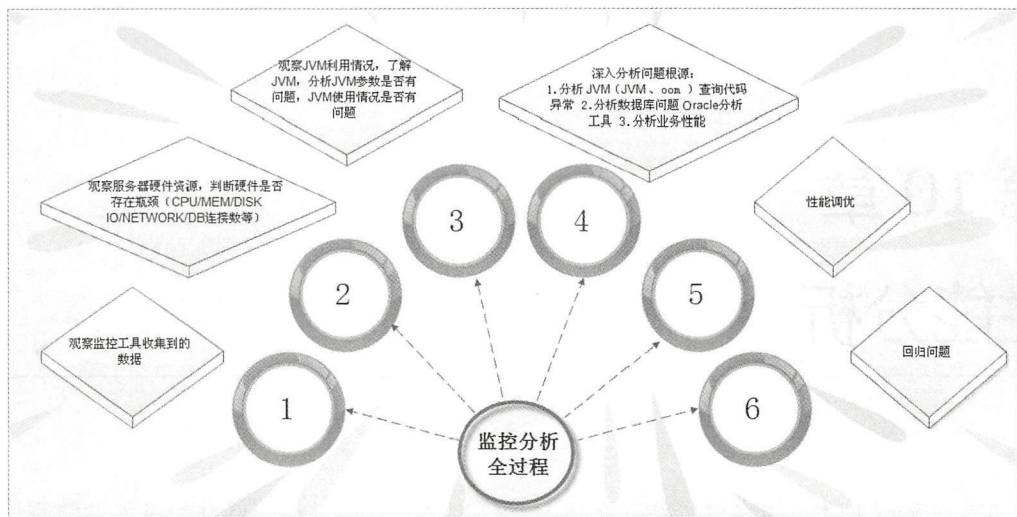


图 10-1 监控分析全过程

通过监控工具对硬件资源和线程数量的分析，可以分析一些简单的性能问题，或者排除一些硬件方面的性能问题。性能分析的过程也是一个解读数据的过程，读懂了数据就能知道问题出在何处。随着经验的积累有助于更早地判断问题的根源，甚至在开发阶段时可以对可能出现的问题点打预防针。本章讲解的内容如下：

- 系统硬件资源监控
- nmon 的使用与分析
- 常见的 Linux 监控命令
- 了解 JVM 的组成和常见问题
- JVM 监控工具的使用
- JVM 问题分析和命令和实战
- SQL 执行计划分析性能问题
- 解读 AWR 报告，了解关键的数据是哪些

10.1 系统硬件资源监控

性能测试时，除了 LoadRunner 和 JMeter 用于收集应用程序的性能指标，还要结合服



务器端的系统性能指标，才能够更准确地评估性能状况。性能问题的第一步就是排除硬件的性能问题，之后才能进行后续的性能问题分析。下面介绍一些常用的系统监控工具和命令。

10.1.1 nmon 工具

nmon 是 IBM 提供的开源的硬件资源监控和分析工具，使用它能够判断性能问题是否由硬件产生。目前，nmon 已经可以在 AIX、Ubuntu、openSUSE、Red Hat、Fedora 等操作系统上稳定运行。nmon 可以说是 AIX 系统中的 topas 工具、Linux 中 top 命令的强化版，它可以在一个屏幕上显示所有重要的性能数据，动态更新，而且不会消耗大量的 CPU 使用率（通常低于 2%）。nmon 主要可以收集 CPU 使用率、内存使用情况、内核统计信息、运行队列信息、磁盘 I/O 情况和网络 I/O 使用情况等。

nmon 除了能实时监控，还能进行长时间的性能数据收集工作。它将数据捕获到一个文本文件中，以表格形式（.csv 格式）存储，可以使用 nmon analyser 分析工具（是一个 Excel 宏文件）对数据进行全方位解析，生成形象具体的硬件性能趋势图。通过图中的变化曲线和性能测试的现象，能准确地定位在长时间性能测试中硬件是不是瓶颈、硬件会不会出现资源利用不足或过剩的现象等。

根据 CPU 的类型和操作系统选择下载相应的 nmon 版本，网址为：<http://nmon.sourceforge.net/pmwiki.php?n=Site.Download>。

本节介绍如何配置使用 nmon，在实例中使用的系统和软件版本如下。

- 操作系统：64 位 redhat5-2.6.18
- nmon 工具：nmon_x86_64_rhel5
- nmon 结果分析工具：nmon_anlayser v46.xls

1. nmon 配置

nmon 的安装配置很简单，按照以下 4 步即可。

(1) nmon 工具是一个独立的二进制文件，将 nmon_x86_64_rhel5 文件以二进制的方式上传到服务器的/usr/bin 目录下，并授予 root 权限。如果没有 root 权限，则可以上传到/home/admin 目录下，使用时到/home/admin 目录下执行 nmon 命令即可。

(2) 修改文件名字为 nmon，便于日后使用，命令如下：



```
# mv nmon_x86_64_rhel5 nmon
```

(3) 修改 nmon 的使用权限为 755 rwxr-xr-x（可读可写可执行），命令如下：

```
# chmod 755 nmon
```

(4) 在对应路径下输入“./nmon”命令，出现如图 10-2 所示的界面，这就表示 nmon 可以正常使用了。

```
nmon-14a-----[H for help]-----Hostname=tombat063-----Refresh= 2secs -----11:14:34-----

          For help type H or ...
          nmon -? - hint
          nmon -h - full

          To start the same way every time
          set the NMON ksh variable

          Use these keys to toggle statistics on/off:
          c = CPU          l = CPU Long-term      - = Faster screen updates
          M = Memory       j = Filesystems         + = Slower screen updates
          d = Disks         n = Network             V = Virtual Memory
          r = Resource     W = NFS                  v = Verbose hints
          h = kernel       t = Top-processes        . = only busy disks/procs
          b = more options  q = Quit
```

图 10-2 nmon 图形界面

2. nmon 实时监控

nmon 可以实时监控硬件资源利用率，默认每 5 秒刷新一次数据。在 nmon 主界面中依次按下字符 c 键、字符 m 键、字符 d 键和字符 t 键，会出现如图 10-3 所示的实时监控界面。它非常直观地显示出当前压力测试下被测系统的整体硬件资源利用情况。显示结果从上而下为 CPU Utilisation、Memory Stats、Disk I/O 和 Top Processes。这里输入的字符键含义如下：

- c (c=CPU)，可以查看 CPU 使用率，如操作系统的 User%、Sys%和 Wait%等。
- m (m=Memory)，可以查看内存使用情况，如内存已使用的容量和空闲的容量等。
- d (d=Disks)，查看磁盘使用情况，如交换率等。
- t (t=Top 命令)，显示和 top 命令一样，包括各个进程的 CPU%、Disk、内存使用情况等。

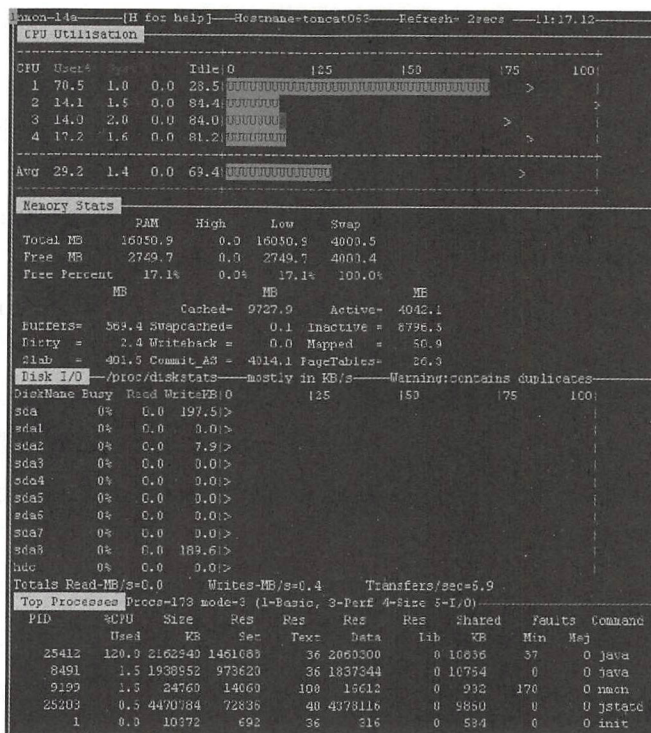


图 10-3 nmon 显示系统整体硬件资源利用情况

除了实例中使用的字符键“c、m、d、t”，还可以使用以下的字符键监视性能状况。

- r (r=Resource)，系统资源视图。
- k (k=Kernel)，内核统计信息。
- l (l=CPU long term)，长期处理器平均使用率视图。
- j (j=Filesystems)，文件系统视图。
- n (n=Network)，网络接口视图。
- N (N=NFS)，网络文件系统视图。

(4) 按字符键 q，可以退出 nmon 界面。

3. nmon 实时收集数据

nmon 除了可以实时监控，还可以收集一段时间内的性能数据，下面介绍收集数据的方法和分析结果。

(1) 在待监控的服务器上执行如下命令：



```
# ./nmon -fT -s 10 -c 360
```

在此命令中，-s 10 代表每 10 秒收集一次性能数据，-c 360 代表收集 360 次。整条命令的意思是每 10 秒钟收集 1 次硬件信息，总共收集 1 小时的数据。对命令中的 10 和 360 做修改就可以收集任何时间的数据了。当然，收集的硬件信息指的是即时的性能数据，而不是历史记录，所以这个命令在性能测试开始时就需要运行，这样才可以收集到从测试开始到结束的所有硬件资源使用信息。

(2) nmon 命令执行完成后，生成一个前缀为机器名，扩展名为.nmon 的文件，如 mycomputer.nmon，然后下载该文件到本地计算机。

(3) 在本地计算机上，打开 nmon 分析工具，如 nmon_anlaysia_v46.xls。打开后，在如图 10-4 所示的 Excel 界面中，单击“Analyze nmon data”按钮，选择上一步下载的 mycomputer.nmon 文件，开始解析性能数据，最后生成服务器硬件资源性能趋势图，如 CPU、内存、磁盘和网络的使用情况。

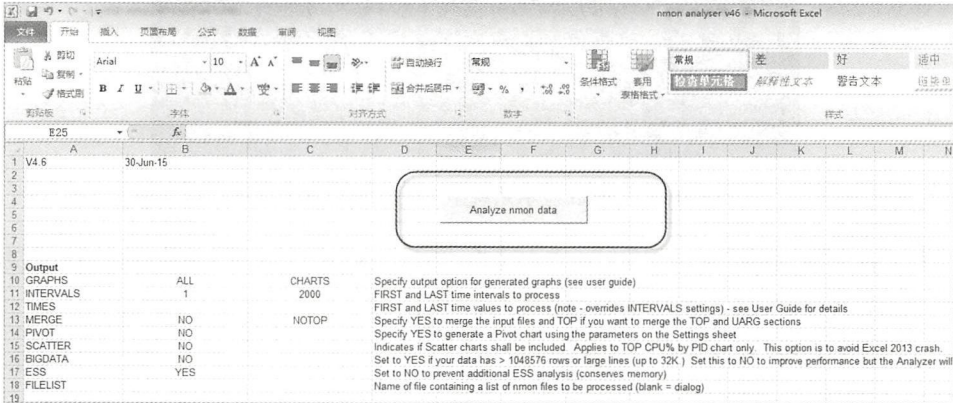


图 10-4 nmon analysis 界面

如图 10-5 所示为 CPU 信息分析截图，显示 User%、Sys%和 Wait%，图中第一条曲线是 Sys%大约为 5%，第二条曲线是 User%大约为 60%，而 Wait%基本为 0，因此没在图中显示，总体上 CPU 状态良好。

如图 10-6 所示为 Memory 信息分析截图，默认显示的是空闲的内存容量，图中显示空闲内存为 2.75GB 左右。

如图 10-7 所示为 DISK 信息分析截图，显示磁盘读写速度和 IO 交换率。

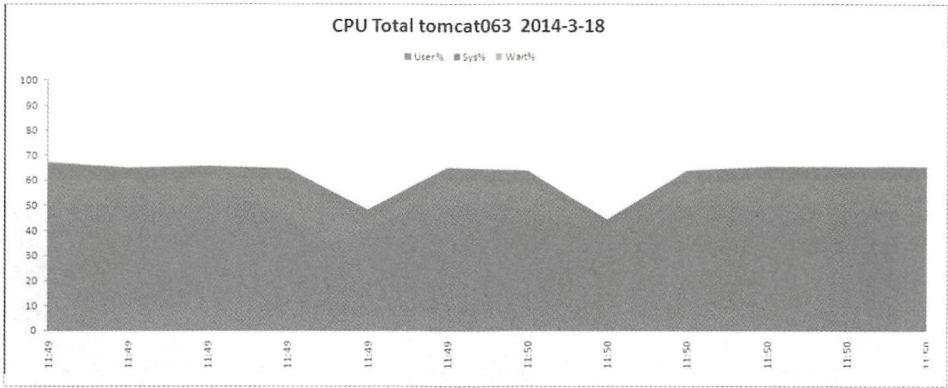


图 10-5 nmon 收集的 CPU 信息分析截图

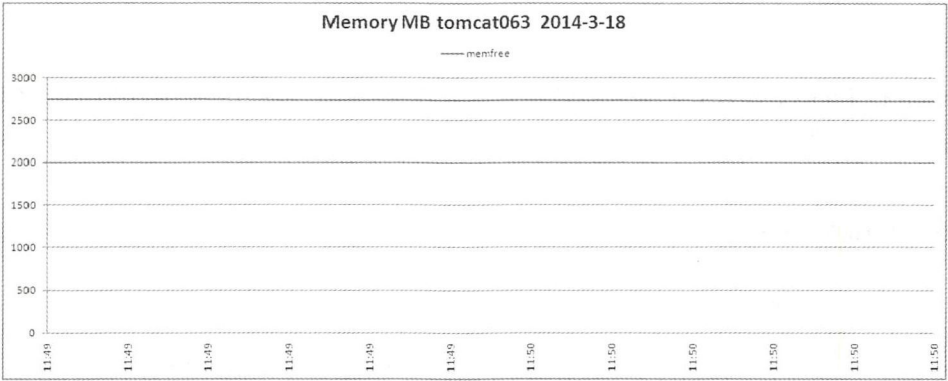


图 10-6 nmon 收集的内存信息分析截图

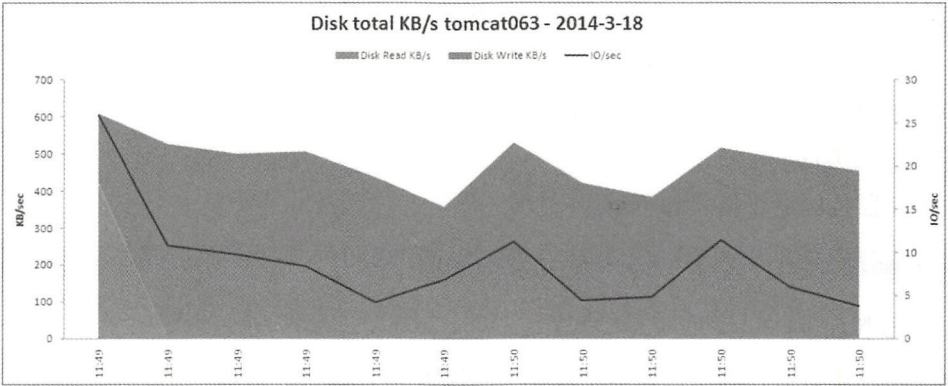


图 10-7 nmon 收集的磁盘信息分析截图



如图 10-8 所示为 NetWork 流量分析截图，显示总的网络读写速度。

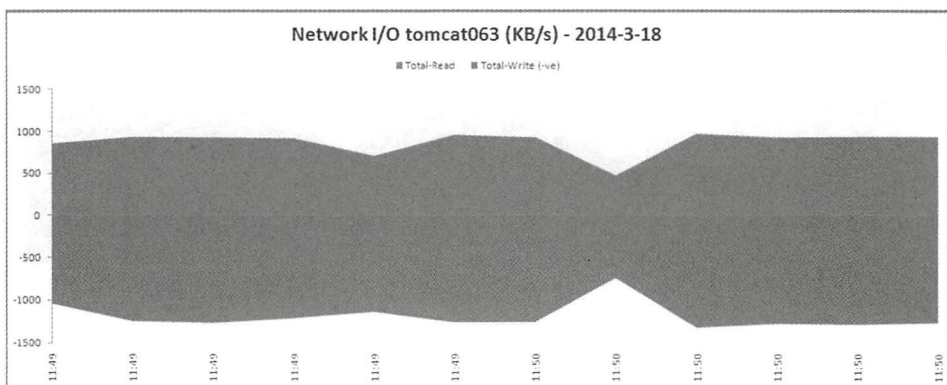


图 10-8 nmon 收集的网络信息分析截图

nmon 在 Linux、UNIX 系统上的硬件监控表现都不错，可以作为 24 小时的服务器性能体检专家。万一服务器宕机了，也能保留当时内存耗尽时的情况。

10.1.2 Linux 系统监控命令

一般主流服务器的操作系统以 Linux、UNIX 之类居多，掌握常用的 Linux 命令有助于监控分析硬件资源的性能问题，下面主要讲解 Linux 监控系统的常用命令。

1. netstat 网络监控

netstat 是一个监控 TCP/IP 网络非常有用的工具。它可以显示路由表、实际的网络连接及每一个网络接口设备的状态信息，也可以显示与 IP、TCP、UDP 和 ICMP 协议相关的统计数据。该命令只有在安装了 TCP/IP 协议后才可以使用。

(1) 使用方法

netstat 监控的命令格式如下：

```
netstat [-a] [-e] [-n] [-s] [-p protocol] [-r] [interval]
```

常见参数如下：

- a (all) 显示所有选项，默认不显示 LISTEN 相关选项。
- t (tcp) 仅显示 TCP 相关选项。
- u (udp) 仅显示 UDP 相关选项。



-n 拒绝显示别名，能显示数字的全部转化成数字。

-l 仅列出 Listen（监听的服务状态）。

-p 建立相关连接的程序名。

-r 显示路由信息，如路由表。

-e 显示扩展信息，如 uid 等。

-s 按各个协议进行统计。

-c 每隔一个固定时间，执行 netstat 命令。

提示：LISTEN 和 LISTENING 的状态只有用 -a 或者 -l 才能看到。

在 netstat 显示的服务列表中，PID 和 ps aux 的进程列表是一样的。如果服务器上有几个 Java 进程同时运行，能够按 PID 找到每个进程就很重要了。建议每台服务器运行服务少一点，必要时可以增加服务器。

（2）netstat 输出结果

netstat 的输出结果分为两个部分。

一个是 Active Internet connections，称为有源 TCP 连接，如图 10-9 所示，其中“Recv-Q”和“Send-Q”是指接收队列和发送队列，一般都应该为 0，如果不是则表示网络包正在队列中堆积。

```
[root@localhost ~]# netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:10051          0.0.0.0:*                LISTEN      15353/zabbix_server
tcp        0      0 0.0.0.0:22             0.0.0.0:*                LISTEN      1053/sshd
tcp        0      0 0.0.0.0:1:25           0.0.0.0:*                LISTEN      1448/master
tcp6       0      0 10.108.12.196:10052    :::*                    LISTEN      15323/java
tcp6       0      0 :::3306                :::*                    LISTEN      27565/mysqld
tcp6       0      0 :::80                  :::*                    LISTEN      18264/httpd
tcp6       0      0 :::22                  :::*                    LISTEN      1053/sshd
tcp6       0      0 :::3000                :::*                    LISTEN      12773/grafana-server
tcp6       0      0 :::1:25                :::*                    LISTEN      1448/master
[root@localhost ~]# netstat -nulp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 0.0.0.0:68             0.0.0.0:*                LISTEN      1470/dhclient
udp        0      0 0.0.0.0:24276          0.0.0.0:*                LISTEN      1470/dhclient
udp6       0      0 :::28631               :::*                    LISTEN      1470/dhclient
```

图 10-9 有源 TCP 连接

另一个是 Active UNIX domain sockets，称为有源 UNIX 域套接口，如图 10-10 所示，它和网络套接字一样，但是只能用于本机通信，性能可以提高一倍。

图中的“Proto”表示连接使用的协议，“RefCnt”表示连接到本套接口上的进程号，“Types”显示套接口的类型，“State”表示套接口当前的状态，“I-Node”表示文件结构的 inode 号，“PID/Program name”表示进程号和程序号，“Path”表示连接到套接口的其他进程使用的路径名。



```
[root@localhost ~]# netstat -nlp
```

Active UNIX domain sockets (only servers)							
Proto	RecvCnt	Flags	Type	State	I-node	PID/Program name	Path
unix	2	ACC	STREAM	LISTENING	14847	682/NetworkManager	/var/run/NetworkManager/private
unix	2	ACC	STREAM	LISTENING	16890	1448/master	public/cleanup
unix	2	ACC	STREAM	LISTENING	16893	1448/master	public/gmgr
unix	2	ACC	STREAM	LISTENING	16919	1448/master	public/flush
unix	2	ACC	STREAM	LISTENING	16934	1448/master	public/showq
unix	2	ACC	STREAM	LISTENING	13583	1/systemd	/var/run/dbus/system_bus_socket
unix	2	ACC	STREAM	LISTENING	11072	1/systemd	/run/lvm/lvmetad.socket
unix	2	ACC	STREAM	LISTENING	16979	682/NetworkManager	/var/run/NetworkManager/private-dhcp
unix	2	ACC	STREAM	LISTENING	6529	1/systemd	/run/systemd/journal/stdout
unix	2	ACC	SEQPACKET	LISTENING	10903	1/systemd	/run/udev/control
unix	2	ACC	STREAM	LISTENING	16910	1448/master	private/defer
unix	2	ACC	STREAM	LISTENING	16913	1448/master	private/trace
unix	2	ACC	STREAM	LISTENING	16916	1448/master	private/verify
unix	2	ACC	STREAM	LISTENING	16922	1448/master	private/verify

图 10-10 有源 UNIX 域套接口

(3) 检查服务器的物理连接数

下面的命令可以检查服务器的 tcp 物理连接数：

```
netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}|egrep 'ESTABLISHED'|awk '{print $2}'
```

其中各参数说明如下：

- netstat -n 表示以数字形式显示地址和端口号。
- /^tcp/ 是正则表达式，表示匹配以 tcp 开头的行记录。
- {++S[\$NF]} END {for(a in S) print a, S[a]} 表示每匹配到一行记录，数组 S[\$NF] 就增加 1，其中 NF 是当前匹配行的最后一个字段，作为数组 S 的元素索引，字段值显示连接状态如“ESTABLISHED”或者“CLOSE_WAIT”等。
- egrep 'ESTABLISHED' 表示只查找“ESTABLISHED”状态的网络连接。
- awk '{print \$2}' 表示打印的第二列。

为达到同样效果，也可以使用如下命令：

```
netstat -nat|awk '/^tcp/|egrep 'ESTABLISHED'|wc -l
```

netstat 还可以监控某端口的网络客户连接数，命令如下：

```
netstat -n | grep tcp | grep 8080 | wc -l
```

(4) 实例：性能测试中 TCP 连接过多的性能问题

在性能测试过程中，首先监控 TIBCO 消息服务器，使用 lsof 命令时发现 TIBCO 进程中打开文件的数量在持续增加，当数量增加到服务器设置的最大限制总数时，导致 TIBCO 没有响应。进一步使用 netstat 监控发现应用服务器的物理连接数也在持续增加，问题根源在于应用程序开启的长连接未正确释放。正常释放长连接后进行回归测试，场景无异常并能稳定运行。通过监控脚本发现，应用服务器物理连接数和 TIBCO 消息服务器打开的文件数均未出现持续增长趋势。


```
ll /proc/8491/task | wc -l
```

```
[root@tomcat063 task]# ls
```

10313	10344	10370	10574	10584	10607	10952	10995	12377	12404	12445	12474	12506	12547	0497	0506	0517	0537
10319	10346	10372	10577	10595	10608	10953	10997	12379	12406	12448	12479	12515	12559	0498	0507	0518	0538
10321	10347	10380	10578	10587	10609	10950	12365	12363	12416	12451	12460	12516	12570	0499	0508	0521	0539
10323	10348	10388	10579	10600	10610	10959	12367	12384	12417	12452	12464	12524	12571	0500	0509	0523	0540
10327	10349	10390	10586	10601	10953	10970	12368	12365	12422	12461	12465	12525	0491	0501	0510	0524	0541
10329	10353	10401	10561	10602	10954	10971	12369	12391	12424	12466	12490	12527	0493	0502	0511	0526	0542
10332	10355	10569	10588	10603	10958	10974	12371	12392	12431	12467	12491	12532	0494	0503	0512	0527	0545
10336	10360	10571	10567	10605	10959	10975	12372	12393	12434	12469	12492	12534	0495	0504	0513	0535	0546
10343	10369	10572	10592	10606	10961	10976	12373	12399	12439	12472	12504	12537	0496	0505	0516	0536	0547

图 10-12 进程 8491 产生的线程

3) 进入某个线程的目录，如 10313，可以查看线程目录信息。

```
cd 10313
```

4) 输入 more status 命令观察线程的详细内容，如图 10-13 所示，可以发现线程 10313 属于线程组 8491（Tgid 一列），而当前线程组有 267 个线程（Threads 一列）。

```
[root@tomcat063 10313]# more status
```

Name:	java
State:	S (sleeping)
SleepAVG:	98%
Tgid:	8491
Pid:	10313
PPid:	1
TracerPid:	0
Uid:	500
Gid:	500
FDSize:	512
Groups:	500 501
VmPeak:	2181568 kB
VmSize:	2078760 kB
VmLck:	0 kB
VmHWM:	1096392 kB
VmRSS:	1089732 kB
VmData:	1977064 kB
VmStk:	88 kB
VmExe:	36 kB
VmLib:	69932 kB
VmPTE:	3080 kB
StaBrk:	5aaffe000 kB
Brk:	606bb000 kB
StaStk:	7ffff6512460 kB
Threads:	267
SigQ:	0/139264
SigPnd:	0000000000000000
ShdPnd:	0000000000000000
SigBlk:	0000000000000004
SigIgn:	0000000000000002
SigCgt:	1000000181005ccd
CapInh:	0000000000000000
CapPrm:	0000000000000000
CapEff:	0000000000000000
Cpus_allowed:	00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000
Mems_allowed:	00000000,00000001

图 10-13 线程的具体内容

(2) 图形化实时监控服务器线程信息

使用 `top` 命令可以图形化监控线程，命令如下，注意一定要区分大小写，输入后显示如图 10-14 所示的界面。

`top-p 8491 H`

```
top - 19:58:42 up 130 days, 16:49, 3 users, load average: 0.01, 0.02, 0.01
Tasks: 267 total, 0 running, 267 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.3%us, 0.4%sy, 0.0%ni, 99.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 16436124k total, 13854212k used, 2581912k free, 585508k buffers
Swap: 4096532k total, 124k used, 4096408k free, 10071748k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8617	oracle	15	0	2030m	1.0g	10m	S	0.7	6.6	1:24.49	java
8491	oracle	20	0	2030m	1.0g	10m	S	0.0	6.6	0:00.00	java
8493	oracle	22	0	2030m	1.0g	10m	S	0.0	6.6	1:17.21	java
8494	oracle	15	0	2030m	1.0g	10m	S	0.0	6.6	0:23.19	java
8495	oracle	16	0	2030m	1.0g	10m	S	0.0	6.6	0:23.20	java
8496	oracle	16	0	2030m	1.0g	10m	S	0.0	6.6	0:23.11	java
8497	oracle	16	0	2030m	1.0g	10m	S	0.0	6.6	0:23.12	java
8498	oracle	15	0	2030m	1.0g	10m	S	0.0	6.6	0:53.45	java
8499	oracle	16	0	2030m	1.0g	10m	S	0.0	6.6	0:01.32	java
8500	oracle	16	0	2030m	1.0g	10m	S	0.0	6.6	0:06.96	java

图 10-14 top 监控线程组

图中显示的“Tasks: 267 total”就是进程 8491 下所有的线程数。如果 Tasks 的总量一直不停上涨，当心有宕机的危险。图中下半部门显示的是进程 8491 下所有线程的状态，如果某个线程长期占据很高的 CPU 或内存使用率，这个线程可能就有问题。

(3) 案例：某系统性能测试 1 小时左右宕机

在某次性能测试中，每次测试进行到 1 小时左右系统就宕机。后来发现线程的总数量一直上涨，最后到达 4000 左右的时候系统宕机。跟踪代码后发现是一个文件句柄没关闭，打开了太多的文件，导致线程不断疯涨，耗尽了所有的内存。

3. pmap: 查看地址空间分布

`pmap` 用于观察系统中指定进程的地址空间分布情况，显示一个目标文件或者链接库文件中的各个段大小，观察是否有占用大量内存的文件，或者是否有打开后没关闭的文件等，非常适合判断本地内存溢出。`pmap` 命令实质上是读取 `/proc/pid/maps` 中的数据。

(1) pmap 用法

`pmap` 的命令格式如下：

```
pmap [-x | -d] [-q] pids...
pmap -V
```

其中命令格式中的常用参数说明如下。

-x: extended, 显示扩展格式。

-d: device, 显示设备格式。

-q: quiet, 不显示头尾行。

-V: show version, 显示版本。

可以使用如下 pmap 命令, 查看文件地址、信息并显示扩展格式:

```
pmap -d 25412 | more
```

或者直接查看 pmap 文件, 命令如下:

```
more /proc/25412/maps
```

如图 10-15 所示是通过 pmap 命令截取的一个虚拟内存快照, 表现为一个虚拟的文件系统, 从左往右各项内容说明如下。

- Address: 内存的起始地址。
- Kbytes: 占用内存的字节数 (KB)。
- Mode: 内存的权限, 包括 read、write、execute、shared、private。
- Offset: 文件偏移。
- Device: 设备名。
- Mapping: 占用内存的文件、[anon] (实际分配的内存) 或者是[stack] (堆栈)。

```
[root@tomcat063 25412]# pmap -d 25412 | more
25412: /opt/jdk1.6.0_29/bin/java -Djava.util.logging.config.file=/
t.jmxremote -Dcom.sun.management.jmxremote.port=6914 -Dcom.sun.manage
f/jmxremote.password -Dcom.sun.management.jmxremote.access.file=./st
Address      Kbytes Mode Offset      Device Mapping
0000000040000000 36 r-x-- 0000000000000000 008:00008 java
0000000040108000 8 rwx-- 0000000000008000 008:00008 java
00000000401c3000 4 ---- 00000000401c3000 000:00000 [ anon ]
00000000401c4000 1024 rwx-- 00000000401c4000 000:00000 [ anon ]
00000000402c4000 12 ---- 00000000402c4000 000:00000 [ anon ]
00000000402c7000 1016 rwx-- 00000000402c7000 000:00000 [ anon ]
000000004046f000 12 ---- 000000004046f000 000:00000 [ anon ]
0000000040472000 1016 rwx-- 0000000040472000 000:00000 [ anon ]
00000000405c2000 12 ---- 00000000405c2000 000:00000 [ anon ]
00000000405c5000 1016 rwx-- 00000000405c5000 000:00000 [ anon ]
00000000406c3000 12 ---- 00000000406c3000 000:00000 [ anon ]
00000000406c6000 1016 rwx-- 00000000406c6000 000:00000 [ anon ]
0000000040854000 12 ---- 0000000040854000 000:00000 [ anon ]
0000000040857000 1016 rwx-- 0000000040857000 000:00000 [ anon ]
000000004095a000 12 ---- 000000004095a000 000:00000 [ anon ]
000000004095d000 1016 rwx-- 000000004095d000 000:00000 [ anon ]
0000000040a5b000 12 ---- 0000000040a5b000 000:00000 [ anon ]
0000000040a5e000 1016 rwx-- 0000000040a5e000 000:00000 [ anon ]
0000000040b5c000 12 ---- 0000000040b5c000 000:00000 [ anon ]
```

图 10-15 虚拟内存快照

(2) 实例之内存溢出分析

在程序执行中，经常会见到下面的本地内存溢出错误：

```
- Exception in thread "main" java.lang.OutOfMemoryError: request <size> bytes for
<reason>. Out of swap space?
- Exception in thread "main" java.lang.OutOfMemoryError: <reason>
- <stack trace>(Native method)
```

如何来分析这个问题呢？这里了解一下 pmap 内存溢出分析的原理：pmap 是根据内存块的分类来区分占用大小的。

- 打开的文件：文件所占用的内存块非常容易辨认，其属性为 r-xs-（可读共享），并且其文件名也会被清楚地显示出来，如图 10-16 所示。

```
00002aaab2c8d000      8 r-xs- 0000000000007000 008:00008 el-api.jar
00002aaab2c8f000     60 r-xs- 00000000000667000 008:00008 charsets.jar
00002aaab2cac000      8 r-xs- 000000000000c000 008:00008 catalina-ant.jar
00002aaab2cae000     40 r-xs- 00000000000b2000 008:00008 tomcat-coyote.jar
00002aaab2cb8000      8 r-xs- 0000000000014000 008:00008 servlet-api.jar
00002aaab2cba000     12 r-xs- 0000000000019000 008:00008 jasper-el.jar
00002aaab2cbd000     64 r-xs- 000000000011d000 008:00008 catalina.jar
00002aaab2ccd000     28 r-xs- 000000000007a000 008:00008 jasper.jar
```

图 10-16 “打开的文件”内存块截图

- 加载的动态库：动态库可以很明显地根据其名称确认，一般动态库以 so 结尾。一个动态库有两个内存块，其中一块属性为 r-x--（可读可执行），为该动态库的代码区；另一块为 rwx--（可读可写可执行），为该动态库的数据区，如图 10-17 所示。

```
00002aaaaaabc000     52 r-x-- 0000000000000000 008:00008 libverify.so
00002aaaaaac9000    1020 ---- 000000000000d000 008:00008 libverify.so
00002aaaaabc8000     12 rwx-- 000000000000c000 008:00008 libverify.so
00002aaaaabcb000    164 r-x-- 0000000000000000 008:00008 libjava.so
00002aaaaabf4000    1020 ---- 0000000000029000 008:00008 libjava.so
00002aaaaacf3000     28 rwx-- 0000000000028000 008:00008 libjava.so
```

图 10-17 “加载的动态库”内存块截图

- 线程栈：线程栈由两块相近的内存块构成，其中一块大小在 4KB~20KB 之间，属性为----（不可读），表示为隔离区，用来防止线程栈内存块被其他线程等侵占；另一块为真正的线程栈，属性为 rwx--（可读可写可执行），如图 10-18 所示。

0000000057d96000	1016	rwX--	0000000057d96000	000:00000	[anon]
0000000058298000	12	-----	0000000058298000	000:00000	[anon]
000000005829b000	1016	rwX--	000000005829b000	000:00000	[anon]
000000005899f000	12	-----	000000005899f000	000:00000	[anon]
00000000589a2000	1016	rwX--	00000000589a2000	000:00000	[anon]
0000000058fa5000	12	-----	0000000058fa5000	000:00000	[anon]
0000000058fa8000	1016	rwX--	0000000058fa8000	000:00000	[anon]
00000000593a9000	12	-----	00000000593a9000	000:00000	[anon]

图 10-18 “线程栈”内存块截图

- JVM 内存：即 JVM 堆内存、永久区等 JVM 控制的内存，其属性为 rwX--（可读可写可执行），并且前后没有隔离区，有时会连续出现，如图 10-19 所示。

00002aaaae842000	15972	rwX--	00002aaaae842000	000:00000	[anon]
00002aaaafbc3000	25184	rwX--	00002aaaafbc3000	000:00000	[anon]
00002aaab1843000	312	rwX--	00002aaab1843000	000:00000	[anon]
00002aaab1891000	456	rwX--	00002aaab1891000	000:00000	[anon]
00002aaab1903000	260	rwX	00002aaab1903000	000:00000	[anon]

图 10-19 “JVM 内存”内存块截图

提示：上面的描述是区分内存块的关键点，通过观察内存块的大小可以判断本地内存溢出是由什么引发的。

10.2 JVM 监控与分析

上一节讲述了硬件资源的监控，在排除硬件问题后，如果 Java 程序仍有性能问题怎么办？Java 程序从面世起就被人诟病性能慢，比起机器优化代码，在 JVM 虚拟机上运行确实要慢些。那么究竟什么是 JVM？又该如何去监控分析 JVM 呢？监控和分析用什么工具不重要，只要掌握了 JVM 问题的本质就能很快判断出 JVM 性能问题的根源。

10.2.1 JVM 基础

JVM，是 Java Virtual Machine（Java 虚拟机）的缩写。在谈论 JVM 之前，先来看一下 Java 程序的执行过程，如图 10-20 所示。Java 源代码（.java）文件被编译器编译为字节码（.class）文件后，由 JVM 的类加载器加载，然后交给执行引擎运行。在运行过程中，JVM 会使用一段空间来存储程序执行期的数据和信息，这段空间被称作为 Runtime Data Area（运

运行时数据区)，也就是常说的 JVM 内存。根据《Java 虚拟机规范》的规定，运行时数据区通常包括这几个部分：程序计数器（Program Counter Register）、Java 栈（VM Stack）、本地方法栈（Native Method Stack）、方法区（Method Area）和堆（Heap）。

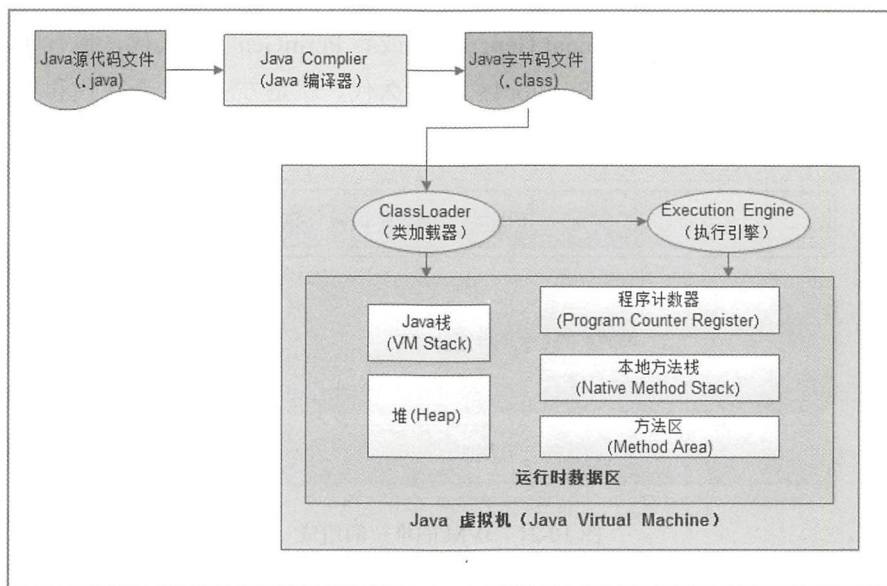


图 10-20 JVM 的组成

对于运行中的 Java 程序而言，其中的每一个线程都有它自己的 PC（程序计数器）。PC 在线程启动时创建，大小是一个字长。它既能持有一个本地指针，也能持有一个返回地址。当线程执行某个 Java 方法时，PC 的内容总是指向下一条指令的“地址”。这里的“地址”可以是一个本地指针，也可以是在方法字节码中相对于该方法起始指令的偏移量。如果该线程正在执行一个本地方法，那么此时 PC 寄存器的值为“undefined”。

本地方法栈是虚拟机使用到的 Native 方法服务，被 Native 方法接口所使用，存储了每个 Native 方法调用的信息。

JVM 内存中的堆只有一个，被所有线程共享，是用来存储对象本身以及数组，而数组引用是存放在 Java 栈中的。Java 的垃圾回收机制会自动处理空间释放的问题，因此这部分空间也是 Java 垃圾收集器（Garbage Collector）需要进行内存管理的主要区域。

方法区也是被线程共享的区域，存储了每个类的信息，包括类的名称、方法、字段、静态变量、常量以及编译器编译后的代码等。

10.2.2 JVM 垃圾回收

JVM 根据 Generation（代）进行垃圾收集，如图 10-21 所示，被分为 Young Generation 或者 New Generation（新生代，也有叫年轻代）、Tenured Generation 或者 Old Generation（年老代，也有叫老年代）、Permanent Generation 或者 PermGen（持久代，也有叫永久代）。其中新生代包括一个 Eden，两个 Survivor 区。“持久代”就是方法区，新生代和年老代都在堆空间内。

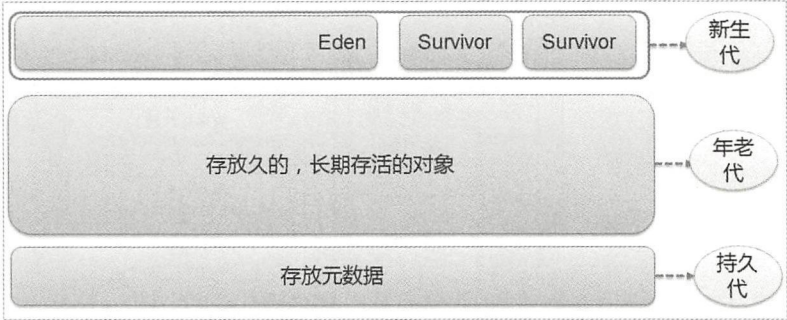


图 10-21 JVM 的堆空间组成

JVM 的垃圾收集器主要包含 3 种回收方式：Minor GC、Major GC 和 Full GC（简称 FGC）。

（1）Minor GC

从新生代（包括 Eden 和 Survivor 区域）回收内存的过程被称为 Minor GC。Survivor 区分为 S1 和 S2 两个区，一般会保持一个区是空的。新建的对象都会放置在 Eden 区中，当 Eden 区已满、无法分配空间给新建对象时，就会发生 Minor GC。此时会将存放在在 Eden 区中的存活对象复制到原先为空的 S1 区，而 Eden 区基本上会被清空。另一个 S2 区的存活对象会被复制到年老代，S2 会被清空。Minor GC 发生后，Eden 区和其中一个 Survivor 区被清空，而另一个 Survivor 区保留着短期的存活对象。

（2）Major GC

从年老代回收内存的过程被称为 Major GC。Major GC 大都是由 Minor GC 触发的。如果 Survivor 区全满，装不下更多来自 Eden 区或另一个 Survivor 区的对象时，就会将有一定年龄的存活对象复制至年老代。当年老代放置不下对象时，就会触发 Major GC。Major GC 的执行代价比 Minor GC 要昂贵很多，速度比 Minor GC 要慢至少 10 倍以上。

(3) Full GC

Full GC 是指清理整个堆空间的过程，包括新生代和年老代。

当垃圾回收出现异常时，会记录 GC 的执行频率和回收内存的日志，并以此为依据判断 JVM 是否有性能问题，以及调整 JVM 参数和减少 GC 频率等。常见的 GC 日志有以下输出参数。

- `-verbose.gc`: 显示垃圾收集的操作内容，包括最忙和最空闲收集行为发生的时间、收集前后的内存大小、收集需要的时间等。
- `-XX:+printGCDetails`: 详细了解垃圾收集中的变化。
- `-XX:+PrintGCTimeStamps`: 了解垃圾收集发生的时间，自 JVM 启动以后以秒计量。
- `-XX:+PrintHeapAtGC`: 了解堆的更详细的信息。
- `-Xloggc:[file]`: 将 GC 信息输出到单独的文件中。

JVM 出现的最常见问题有内存泄露和内存溢出。

内存泄漏（Memory Leak）是指程序在运行过程中动态申请的内存空间不再使用后没有及时得到释放，从而导致内存无限增长。这可能是算法问题或新建对象过多没有释放。

内存溢出（Out Of Memory, OOM）是指用户在对其数据缓冲区操作时，超过了其缓冲区的边界，尤其是对缓冲区写操作时，缓冲区的溢出很可能导致程序的异常，也就是分配的内存不足以放下数据项。

内存泄漏的堆积，会耗尽所有内存，最终导致内存溢出。内存中不同部分的溢出，有不同的表现，定位方法也是不同的。内存溢出的类型划分如表 10-1 所示。

表 10-1 内存溢出类型划分

对应存放区域	内存溢出类型	常见错误
Java 堆栈（Heap），分为新生代和年老代	堆内存溢出	OutOfMemoryError: Java heap space
方法区，也是永久区（PermGen）	永久区内存溢出	OutOfMemoryError: PermGen space
Java 栈	本地线程资源不足	OutOfMemoryError: unable to create new native thread 或者 StackOverflowError

不同的内存溢出有不同的判断方法，大致方法如图 10-22 所示。首先打印堆映射，然后识别 OutOfMemory 报错的类型，如 JVM 堆内存溢出、永久区内存溢出、本地内存溢出或是本地线程资源不足。确定 OOM 类型后，可以查看对应的代码块，对程序进行优化。

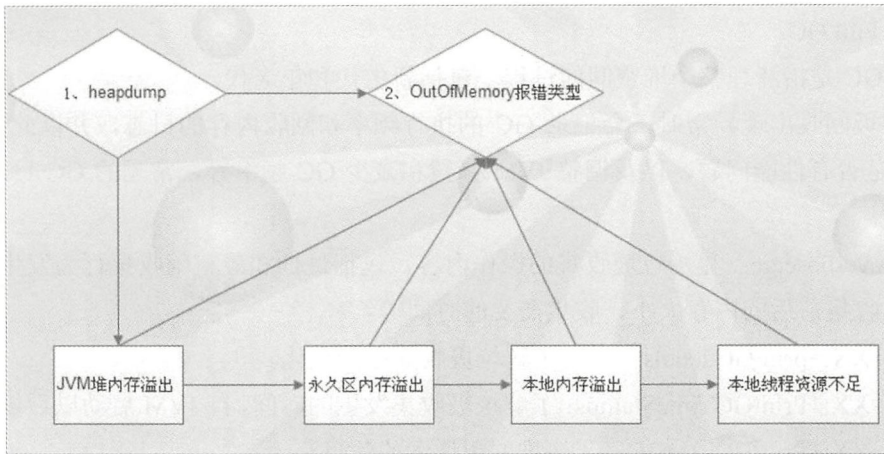


图 10-22 内存溢出的判断方法

导致 OutOfMemoryError 异常的常见原因有以下几种，以供读者参考。

- JVM 启动参数的内存值设定过小，造成堆溢出。
- 一次性加载数据量过大，比如声明对象分配的空间过大。
- 集合类引用对象后，未能及时清空，使得 JVM 内存不能回收。
- 代码中存在死循环或产生过多重复的对象实体。
- 线程池过小，不能再新建更多的原生线程（Native Thread）。
- 交换空间过小。

10.2.3 常见 JVM 命令

为了更好地监控检测 JVM 状态，及时发现内存泄漏和内存溢出问题，下面介绍几个常用的 JVM 命令，结合多个 JVM 命令，可以帮助解决 JVM 的性能问题。

1. lsof：列出系统打开文件

lsof（listopenfiles）是一个列出当前系统打开文件的工具。在 Linux 环境下，任何事物都以文件的形式存在，通过文件不仅仅可以访问常规数据，还可以访问网络连接和硬件、监控文件 IO。由于 lsof 需要访问核心内存和各种形式的文件，所以需要 root 用户执行。

lsof 命令的使用范围非常广泛，能够查看谁在使用文件系统、恢复删除的文件、判断被打开的文件数目是否超过系统的最大文件打开数目设置，同时还能进一步判断本地 IO 是否有问题，是否会产生本地内存溢出。

(1) lsof 用法

lsof 的命令格式如下：

```
lsof [options] filename
```

其中，options 中的各选项说明如下。

- -a: 列出打开文件存在的进程。
- -c<进程名>: 列出指定进程所打开的文件。
- -g: 列出 GID 号的进程详情。
- -d<文件号>: 列出占用该文件号的进程。
- +d<目录>: 列出目录下被打开的文件。
- +D<目录>: 递归列出目录下被打开的文件。
- -n<目录>: 列出使用 NFS 的文件。
- -i<条件>: 列出符合条件的进程。
- -p<进程号>: 列出指定进程号所打开的文件。
- -u: 列出 UID 号的进程详情。
- -h: 显示帮助信息。
- -v: 显示版本信息。

如图 10-23 所示是 lsof 常用方法，列出了 Java 命令下访问的所有文件。

```
lsof | grep -E 'java|COMMAND| more
```

此命令中输出各列信息的说明如下。

- COMMAND: 进程的名称。
- PID: 进程标识符。
- USER: 进程所有者。
- FD: 文件描述符，应用程序通过文件描述符识别该文件，如 cwd、txt 等。
- TYPE: 文件类型，如 DIR、REG 等。
- DEVICE: 指定磁盘的名称。
- SIZE: 文件的大小。
- NODE: 索引节点（文件在磁盘上的标识）。
- NAME: 打开文件的确切名称。


```
[root@tomcat063 oracle]# lsof | grep -E 'java|COMMAND' | more
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
java	8491	oracle	cwd	DIR	8,8	4096	60950030	/opt/oracle/tomcat/t-1/bin
java	8491	oracle	rtd	DIR	8,2	4096	2	/
java	8491	oracle	txt	REG	8,8	50794	61212339	/opt/jdk1.6.0_29/bin/java
java	8491	oracle	mem	REG	8,2	139504	194636	/lib64/ld-2.5.so
java	8491	oracle	mem	REG	8,2	1722304	194637	/lib64/libc-2.5.so
java	8491	oracle	mem	REG	8,2	23360	194638	/lib64/libdl-2.5.so
java	8491	oracle	mem	REG	8,2	615136	194642	/lib64/libm-2.5.so
java	8491	oracle	mem	REG	8,2	145824	194645	/lib64/libpthread-2.5.so
java	8491	oracle	mem	REG	8,2	53448	194646	/lib64/librt-2.5.so
java	8491	oracle	mem	REG	8,2	114352	194657	/lib64/libnsl-2.5.so
java	8491	oracle	mem	REG	8,2	92736	194659	/lib64/libresolv-2.5.so
java	8491	oracle	mem	REG	8,8	22705	60950035	/opt/oracle/tomcat/t-1/bin/bootstrap.jar
java	8491	oracle	mem	REG	8,8	24172	60950038	/opt/oracle/tomcat/t-1/bin/commons-daemon.jar
java	8491	oracle	mem	REG	8,8	32277	60950046	/opt/oracle/tomcat/t-1/bin/tomcat-juli.jar
java	8491	oracle	mem	REG	8,8	76690	60949994	/opt/oracle/tomcat/t-1/lib/jsp-api.jar
java	8491	oracle	mem	REG	8,8	88256	61213162	/opt/jdk1.6.0_29/jre/lib/jce.jar
java	8491	oracle	mem	REG	8,8	231940	61212873	/opt/jdk1.6.0_29/jre/lib/ext/sunpkcs11.jar
java	8491	oracle	mem	REG	8,8	66580	61212983	/opt/jdk1.6.0_29/jre/lib/amd64/libverify.so
java	8491	oracle	mem	REG	8,8	234440	61213001	/opt/jdk1.6.0_29/jre/lib/amd64/libjava.so
java	8491	oracle	mem	REG	8,8	295213	61213005	/opt/jdk1.6.0_29/jre/lib/amd64/libjvmp.so
java	8491	oracle	mem	REG	8,8	15963	61212979	/opt/jdk1.6.0_29/jre/lib/amd64/librpt.so
java	8491	oracle	mem	REG	8,2	56450480	71395	/usr/lib/locale/locale-archive
java	8491	oracle	mem	REG	8,6	32768	864869	/tmp/hsperfdata_oracle/8491
java	8491	oracle	mem	REG	8,8	633975	61213045	/opt/jdk1.6.0_29/jre/lib/jsse.jar
java	8491	oracle	mem	REG	8,2	53880	194523	/lib64/libnss_files-2.5.so
java	8491	oracle	mem	REG	8,8	92253	61212988	/opt/jdk1.6.0_29/jre/lib/amd64/libzip.so
java	8491	oracle	mem	REG	8,8	52185555	61215935	/opt/jdk1.6.0_29/jre/lib/rt.jar
java	8491	oracle	mem	REG	8,8	21974	61213013	/opt/jdk1.6.0_29/jre/lib/amd64/libdt_socket.so
java	8491	oracle	mem	REG	8,8	39481	61213024	/opt/jdk1.6.0_29/jre/lib/amd64/libmanagement.so
java	8491	oracle	mem	REG	8,8	113350	61212977	/opt/jdk1.6.0_29/jre/lib/amd64/libnet.so
java	8491	oracle	mem	REG	8,8	170239	61212874	/opt/jdk1.6.0_29/jre/lib/ext/sunjce_provider.jar
java	8491	oracle	mem	REG	8,2	23736	194521	/lib64/libnss_dns-2.5.so

图 10-23 lsof 查看 Java 进程打开的文件

(2) 常用命令语句

- 查看所有进程占用的文件句柄数，命令如下：

```
lsof -n|awk '{print $2}'|sort|uniq -c |sort -nr|more
```

- 统计所有 Java 进程占用的文件句柄数，命令如下：

```
lsof -n|awk '{print $2}'|grep -E ""ps -ef|grep java | grep -v grep | awk '{print $2}'|sort|uniq -c |sort -nr|more
```

(3) 实例一：查看 jar 文件所占内存

```
lsof | awk '{if($0~/jar/){print $0}}'
```

打开的 jar 文件所占用的内存属于本地内存，如果程序打开的文件太多，或者打开了某个较大的文件，则可能导致本地内存溢出。此时需要实时地监控文件，可以每隔一段时间执行一次 lsof 命令，如图 10-24 所示。

```
[root@tomcat063 bin]# lsprof | awk '{if($0~/jar/){print $0}}'
```

java	8491	oracle	mem	REG	8,8	22705	60950035	/opt/oracle/tomcat/t-1/bin/bootstrap.jar
java	8491	oracle	mem	REG	8,8	24172	60950038	/opt/oracle/tomcat/t-1/bin/commons-daemon.jar
java	8491	oracle	mem	REG	8,8	32277	60950046	/opt/oracle/tomcat/t-1/bin/tomcat-juli.jar
java	8491	oracle	mem	REG	8,8	76690	60949994	/opt/oracle/tomcat/t-1/lib/jsp-api.jar
java	8491	oracle	mem	REG	8,8	88256	61213162	/opt/jdk1.6.0_29/jre/lib/jce.jar
java	8491	oracle	mem	REG	8,8	231940	61212873	/opt/jdk1.6.0_29/jre/lib/ext/sunpkcs11.jar
java	8491	oracle	mem	REG	8,8	633975	61213045	/opt/jdk1.6.0_29/jre/lib/jsse.jar
java	8491	oracle	mem	REG	8,8	52185555	61215935	/opt/jdk1.6.0_29/jre/lib/rt.jar
java	8491	oracle	mem	REG	8,8	170239	61212874	/opt/jdk1.6.0_29/jre/lib/ext/sunjce_provider.jar
java	8491	oracle	mem	REG	8,8	54982	60950004	/opt/oracle/tomcat/t-1/lib/tomcat-i18n-ja.jar
java	8491	oracle	mem	REG	8,8	33332	60950008	/opt/oracle/tomcat/t-1/lib/el-api.jar
java	8491	oracle	mem	REG	8,8	6773558	61213053	/opt/jdk1.6.0_29/jre/lib/charsets.jar
java	8491	oracle	mem	REG	8,8	54564	60950002	/opt/oracle/tomcat/t-1/lib/catalina-ant.jar

图 10-24 lsprof 分析截图

(4) 实例二：定时搜集系统信息

通过 Linux 的定时任务（cron）可以方便地定时搜集 pmap 和 lsprof 等系统信息。对比不同时段的信息，找出变化或增长频繁的部分，可以帮助锁定问题的原因，具体步骤如下。

- 1) 使用业务用户（admin）或者 root 用户登录 Linux，输入 “crontab -e”。
- 2) 单击 “i” 键进入插入模式，向 crontab 文件中输入任务内容：

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * pmap 8491 > /tmp/pmap_8491_`date +%Y%m%d%H%M%S`
.txt
0,5,10,15,20,25,30,35,40,45,50,55 * * * * lsprof -p 8491 > /tmp/lsprof_`date +%Y%m%d%H%M%S`.txt
```

在输入的任务内容中，前面用逗号分隔开的一串数字代表分钟，*号分别代表小时、日期、月份和星期。实例中每隔 5 分钟就会输 pmap 和 lsprof 的信息，然后分别存入 /tmp/pmap_<pid>_xxxx.txt 和 /tmp/lsprof_<pid>_xxxx.txt 文件中。文件名中的 xxx 是根据运行时的时间而变化的。

2. jstat：监视堆和垃圾回收状况

jstat 是 JDK 自带的一个轻量级工具。全称为 “Java Virtual Machine Statistics Monitoring Tool”，是 JVM 的分析监控工具。它位于 JDK 安装目录的 bin 目录下，主要利用 JVM 内建的命令对 Java 应用程序的资源 and 性能进行实时的命令行监控，包括监视 JVM 内存内的各种堆和非堆的大小及其内存使用量，监控堆大小和垃圾回收状况等。

(1) jstat 用法

jstat 命令格式如下：

```
jstat-<option>[-t][-h<lines>]<pid>[<interval>[<count>]]
```

其中，option 包括以下选项。

质量全面管控：从项目管理到容灾测试

- **-class**: 显示加载 class 的数量及所占空间等信息。
- **-compiler**: 显示 VM 实时编译的统计信息，包括编译任务执行的数量、编译任务失败的数量、变为无效的编译任务数量、执行编译任务所花费的时间等。
- **-gc**: 可以显示 gc 的信息，查看 gc 的次数及时间。其中最后 5 项分别是 younggc 的次数、younggc 的时间、fullgc 的次数、fullgc 的时间和 gc 的总时间。
- **-gccapacity**: 显示 VM 内存中三代 (young、old、perm) 对象的使用和占用情况，如 PGCMN 显示的是 perm 的内存最小使用量，PGCMX 显示的是 perm 的内存最大使用量，PGC 是当前新生成的 perm 内存占用量，PC 是当前 perm 内存占用量，OC 是年老代的内存占用量。
- **-gcutil**: 显示 gc 信息统计。
- **-gccause**: 该选项和-gcutil 选项一样都是显示 gc 统计信息，该选项还包括了最后 gc 事件和当前 gc 事件。
- **-gcnew**: 显示 new 对象的信息。
- **-gcnewcapacity**: 显示 new 对象的信息及其占用量。
- **-gcold**: 显示 old 对象的信息。
- **-gcoldcapacity**: 显示 old 对象的信息及其占用量。
- **-gcpermcapacity**: 显示 perm 对象的信息及其占用量。
- **-printcompilation**: 显示当前 VM 执行的信息。

命令格式中的 pid 就是 jps 查看到的进程号；-t 显示第一列为时间戳；-h n 表示每 n 行显示标题；interval 是时间间隔，单位为毫秒；count 就是需要查看的次数。

(2) 实例一：收集垃圾回收信息

目前在项目中 jstat 的 gcutil 选项使用最多，可以观察 FullGC 的次数，再结合服务器总内存使用情况来判断 JVM 参数是否合适以及是否会出现内存泄露或内存溢出的现象。具体使用参考如下命令，<pid>以实际进程号替代。

```
nohup jstat -gcutil <pid> time > filename &
```

如下实例中的命令可以收集 Java 进程的垃圾回收信息，按照一定的时间间隔抽样，每次显示一行信息，结果显示如图 10-25 所示。

```
nohup jstat -gcutil `ps -ef|grep java | grep -v grep | awk '{print $2}'` 10000 > test &
```


此实例命令中的 `ps -ef | grep java | grep -v grep | awk '{print $2}'` 获得 Java 的 PID, 不含 `grep` 命令自身的 PID; 命令中的 10000 指的是 10 秒; `nohup` 和 `&` 结合, 使得 `jstat` 命令在后台运行而不影响其他命令, 输出的结果会存入 `test` 文件中。

S0	S1	E	O	P	YGC	YGCT	FGC	FGCT	GCT
0.00	0.00	5.32	5.55	43.01	711	5.705	332	11.325	17.029
0.00	0.00	5.32	5.55	43.01	711	5.705	332	11.325	17.029
0.00	0.00	5.32	5.55	43.01	711	5.705	332	11.325	17.029
0.00	0.00	5.32	5.55	43.01	711	5.705	332	11.325	17.029
0.00	0.00	5.32	5.55	43.01	711	5.705	332	11.325	17.029
0.00	0.00	5.32	5.55	43.01	711	5.705	332	11.325	17.029

图 10-25 jstat 结果显示垃圾回收信息

这里介绍一下输出的 GC 日志的各字段含义。

- S0: 新生代中第一个 Survivor (幸存区), 已使用的占当前容量百分比。
- S1: 新生代中第二个 Survivor (幸存区), 已使用的占当前容量百分比。
- E: 新生代中 Eden (伊甸园) 已使用的占当前容量百分比。
- O: 年老代已使用的占当前容量百分比。
- P: 持久代已使用的占当前容量百分比。
- YGC: 从应用程序启动到采样时新生代中 GC 次数。
- YGCT: 从应用程序启动到采样时新生代中 GC 所用时间 (s)。
- FGC: 从应用程序启动到采样时年老代 (全 GC) GC 次数。如果 FGC 过多, 有必要调整一下 JVM 参数。
- FGCT: 从应用程序启动到采样时年老代 (全 GC) GC 所用时间 (s)。
- GCT: 从应用程序启动到采样时 GC 用的总时间 (s)。

提示: 如果 JDK 版本是 JDK8, P 字段会被 M 字段替换, 如图 10-26 所示, 是指元空间 (Metaspace) 已使用的占当前容量百分比。

[root@localhost temp]# jstat -gcutil 2885 1000 10										
S0	S1	E	O	M	CCS	YGC	YGCT	FGC	FGCT	GCT
0.00	77.59	14.33	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	14.33	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	14.76	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	14.76	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	15.99	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	15.99	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	15.99	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	16.01	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	16.01	30.84	97.79	95.20	133	1.125	1	0.050	1.175
0.00	77.59	16.01	30.84	97.79	95.20	133	1.125	1	0.050	1.175

图 10-26 jstat 结果显示 FGC 信息

(3) 实例二: 分析 JVM 内存使用情况

在命令行中输入如下命令收集垃圾回收信息:

质量全面管控：从项目管理到容灾测试

```
jstat -gcutil 2885 3000 > JVM.csv &
```

该命令中“2885”是 Java 进程号，“3000”是抓取的时间间隔（ms），“JVM.csv”表示写入文件，“&”表示在后台运行，其产生的 GC 日志如图 10-27 所示。

S0	S1	E	O	P	YGC	YGCT	FGC	FGCT	GCT
4.24	0.00	5.03	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	15.31	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	24.44	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	33.69	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	41.74	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	48.60	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	51.71	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	56.08	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	59.09	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	66.58	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	70.92	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	82.60	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	85.70	70.45	49.27	15336	244.630	17	5.902	250.532
4.24	0.00	89.36	70.45	49.27	15336	244.630	17	5.902	250.532
0.00	4.37	0.00	70.49	49.27	15337	244.655	17	5.902	250.557
0.00	4.37	10.70	70.49	49.27	15337	244.655	17	5.902	250.557
0.00	4.37	22.25	70.49	49.27	15337	244.655	17	5.902	250.557

图 10-27 显示的 GC 日志

把生成的 CSV 文件“JVM.csv”下载到本地计算机，用 MS Excel 工具对前 5 列数据进行处理，生成如图 10-28 所示的图表。可以看到 JVM 正常运行时，新生代（S0，S1）回收时间短、老生代（O）回收正常、持久代（P）无变化、峰值和谷底没有偏差、无内存泄漏。

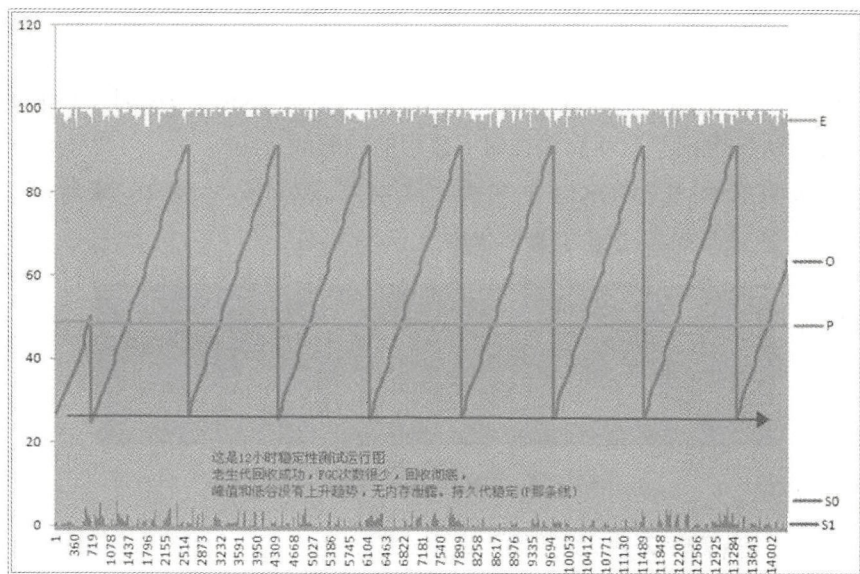


图 10-28 正常运行时 JVM 内存使用情况

对比上图中正常的 JVM 内存情况，如图 10-29 所示的就是有内存泄露的 JVM 内存使用情况。其中第一条线 OC 和第二条线 OU 在逐渐地增加，达到最高值后没有回收减少，表明老生代（OC，OU）在不停增加，而且不能被回收，存在内存泄漏。发现这个内存泄漏问题后，可以用 JProfiler 或者 Yourkit Java Profiler 工具再次深入分析确定哪个类和哪个函数有问题。

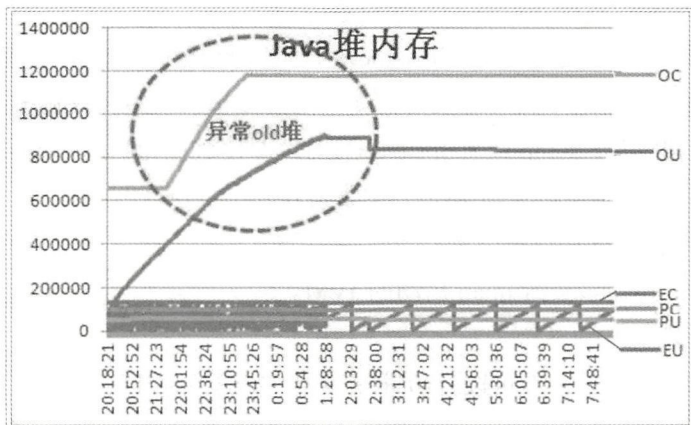


图 10-29 有内存泄露的 JVM 内存使用情况

3. jmap: 获取堆快照

jmap 也是 JDK 自带的一个小工具，同样位于 JDK 的 bin 目录下面。jmap 命令可以获得运行中的 JVM 的堆快照而不会造成测试中的系统宕机。通过堆快照可以分析堆的利用情况，排查堆内和堆外的内存泄露或内存溢出问题，检查哪些对象的创建活动严重影响性能，检查系统中最多的对象和各种对象所占内存的大小等。对备份出来的内存映射信息进一步分析，还可以查找出可能导致内存溢出的类或方法。

(1) jmap 用法

jmap 命令的常用格式如下：

```
jmap [ option ] pid
jmap [ option ] executable core
jmap [ option ] [server-id@]remote-hostname-or-IP
```

其中 pid 是需要被打印配相信息的 Java 进程号，option 包括以下选项。

- -heap: 打印堆信息、GC 使用的算法、堆配置和使用情况。



质量全面管控：从项目管理到容灾测试

- **-histo[:live]:** 打印每个 class 的实例数目、内存占用和类全名信息。JVM 的内部类名字会加上前缀 “*”。假如指定了 live 选项，则只统计活的对象。
- **-clstats:** 打印类加载 ClassLoader 和 JVM 持久代的信息，包含每个 ClassLoader 的名字、活动情况、地址、父 ClassLoader 和加载的 Class 数量，也会打印出内部字符串的数量和占用的内存数。
- **-finalizerinfo:** 打印正等候回收的对象信息。
- **-dump[:live,]format=b,file=<filename>:** 使用 hprof 二进制形式，输出 JVM 的堆内容到文件。live 选项是可选的，假如指定 live 选项，那么只输出活的对象到文件。
- **-F:** 当结合使用 -dump 或者 -histo 参数时，即使进程无响应，也会强制打印堆信息或类加载实例信息。此时 live 子参数无效。
- **-h|-help:** 打印辅助信息。
- **-J<flag>:** 传递参数给 jmap 启动的 JVM。

(2) jmap 常用命令

本例中使用如下 jmap 命令可以打印进程号为 “2883” 的对象数量和内存占用大小，然后存入 pid.log 文件中，命令执行后得到如图 10-30 所示的结果。

```
jmap -histo 2883> pid.log
```

```

169:      21      504 java.util.Locale
170:       9      504 java.util.Properties
171:       9      504 sun.rmi.transport.tcp.TCPCConnection
172:      10     480 java.util.concurrent.ThreadPoolExecutor$Worker
173:      29     464 java.util.concurrent.locks.ReentrantLock
174:      19     456 java.util.concurrent.CopyOnWriteArrayList
175:      19     456 java.util.Locale$LocaleKey
176:      19     448 [Ljava.security.ProtectionDomain;
177:      18     432 sun.reflect.generics.scope.ClassScope
178:      18     432 sun.reflect.generics.tree.ClassSignature
179:      18     432 java.text.DateFormat$Field
180:       3     432 java.text.DecimalFormat
181:      18     432 java.lang.RuntimePermission
182:      18     432 sun.security.action.GetPropertyAction
183:      18     432 sun.reflect.generics.factory.CoreReflectionFactory
184:      18     424 [Lsun.reflect.generics.tree.FormalTypeParameter;

```

图 10-30 jmap 输出的 Heap Dump 信息

如果要将进程号为 “2883” 的堆内存使用的详细情况输出到 heap.hprof 文件中，可以使用如下命令：

```
jmap -dump:format=b,file=/home/test/heap.hprof 2883
```

其中 /home/test/heap.hprof 是打印出来的二进制内存映像文件，可以在堆内存分析工具，如 MAT、JConsole 中进行离线分析。



(3) 实例

以下是 jmap 打印的堆信息，列出了当前各堆的大小和 GC 算法等信息。在实例中，添加注释##以助于理解。本实例中的 JDK 版本为 1.8，所以会看到 Metaspace（元空间）替代了 Permspace（持久空间）。

```
[root@localhost temp]# jmap -heap 2885
Attaching to process ID 2885, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 25.77-b03

using thread-local object allocation.
Mark Sweep Compact GC    ##GC 算法

Heap Configuration:    ##堆配置情况
  MinHeapFreeRatio      = 40    ##最小堆使用比例
  MaxHeapFreeRatio      = 70    ##最大堆使用比例
  MaxHeapSize           = 1073741824 (1024.0MB)  ##最大堆空间大小
  NewSize               = 38404096 (36.625MB)   ##新生代分配大小
  MaxNewSize            = 357892096 (341.3125MB) ##最大新生代可分配大小
  OldSize               = 76939264 (73.375MB)    ##老生代分配大小
  NewRatio              = 2##新生代比例
  SurvivorRatio         = 8##新生代与 Survivor 的比例
  MetaspaceSize         = 21807104 (20.796875MB) ##元空间大小，自 JDK 8 起，元空间替代永久代。
  CompressedClassSpaceSize = 1073741824 (1024.0MB) ##压缩类空间大小
  MaxMetaspaceSize      = 17592186044415 MB      ##最大元空间可分配大小
  G1HeapRegionSize      = 0 (0.0MB)              ##G1 堆区域大小

Heap Usage: ##堆使用情况
New Generation (Eden + 1 Survivor Space): ##新生代分配大小，包括伊甸代和一个存活代
  capacity = 34734080 (33.125MB)##新生代空间大小
  used     = 24119824 (23.002456665039062MB)      ##已使用空间大小
  free     = 10614256 (10.122543334960938MB)##剩余空间大小
  69.44137861143868% used##已使用空间比例
Eden Space:##伊甸代空间
  capacity = 30932992 (29.5MB)##伊甸代空间大小
  used     = 21176464 (20.195449829101562MB)      ##已使用空间大小
  free     = 9756528 (9.304550170898438MB)        ##剩余空间大小
```




质量全面管控：从项目管理到容灾测试

```

68.45915196305614% used##已使用空间比例
From Space:##存活代 From 空间
  capacity = 3801088 (3.625MB)##From 空间（S1）大小
  used      = 2943360 (2.8070068359375MB)          ##已使用空间大小
  free      = 857728 (0.8179931640625MB)          ##剩余空间大小
  77.43467133620689% used                          ##已使用空间比例
To Space:##存活代 To 空间
  capacity = 3801088 (3.625MB)##To 空间（S2）大小
  used      = 0 (0.0MB)                            ##已使用空间大小
  free      = 3801088 (3.625MB)                    ##剩余空间大小
  0.0% used##已使用空间比例
tenured generation:##年老代空间
  capacity = 76939264 (73.375MB)##年老代空间大小
  used      = 23723896 (22.62487030029297MB)##已使用空间大小
  free      = 53215368 (50.75012969970703MB)      ##剩余空间大小
  30.834576218457197% used##已使用空间比例

15314 interned Strings occupying 1950632 bytes.

```

4. jstack: 打印线程栈

jstack 也是 JDK 自带的一个小工具，同样位于 JDK 的 bin 目录下面，可以打印出运行中的 JVM 线程堆栈。如果是在 64 位机器上，则需要指定选项“-J-d64”。

(1) jstack 用法

jstack 命令的常用格式如下：

```

jstack [ option ] pid
jstack [ option ] executable core
jstack [ option ] [server-id@]remote-hostname-or-IP

```

其中，pid 需要被打印配相信息的 Java 进程号，options 中的各选项说明如下。

- -F: 当'jstack [-l] pid'没有响应时强制打印栈信息。
- -l: 长列表，打印关于锁的附加信息。
- -m: 混合模式，打印 Java 和 native C/C++框架的所有栈信息。
- -h|-help: 打印帮助信息。

(2) 命令常用语句

使用 jstack 命令存储某个进程的线程信息，命令如下，其中 PID 为进程号，pid.dmp 为存储堆栈信息的文件名称。



```
jstack <PID>> pid.dmp
```

(3) 实例

在实际分析过程中，往往一次线程堆栈信息还不足以确认问题，建议产生 3 次堆栈信息。如果每次堆栈都指向同一个线程，才能确定问题的根源。以下实例中，jstack 打印进程 15323 的线程堆栈信息。

```
[root@localhost zabbix]# jstack 15323 >15323.pid
"RMI RenewClean-[10.108.12.142:52645]" daemon prio=10 tid=0x00007fd378081000 nid=0x1399 in
Object.wait() [0x00007fd39c99c000]
    java.lang.Thread.State: TIMED_WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x00000000de018770> (a java.lang.ref.ReferenceQueue$Lock)
            at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:135)
            - locked <0x00000000de018770> (a java.lang.ref.ReferenceQueue$Lock)
            at
sun.rmi.transport.DGCClient$EndpointEntry$RenewCleanThread.run(DGCClient.java:535)
        at java.lang.Thread.run(Thread.java:745)
```

从打印的线程栈信息中，读者能得到以下信息。

- nid: NativeID，线程号的 16 进制，此处为 0x1399。
- 线程的状态：TIMED_WAITING。
- 线程的调用栈：记录线程运行中的函数调用（或者也叫函数执行顺序），每一个线程都有一个独自的调用栈。
- 线程的当前锁住的资源：<0x00000000de018770>。

对照 jstack 命令得到的线程号，使用如下的 ps 命令可以列出线程的 CPU 使用率等情况，并能够知道 CPU 使用最高的线程执行到哪里。输出结果如图 10-31 所示。

```
ps -eL -o pid,%cpu,lwp|grep -i 15323
```

```
[root@localhost http-2.0.0]# ps -eL -o pid,%cpu,lwp|grep -i 15323
15323 0.0 15323
15323 0.0 15324
15323 0.0 15325
15323 0.0 15326
15323 0.0 15327
15323 0.0 15328
15323 0.0 15329
15323 0.0 15330
15323 0.0 15331
```

图 10-31 查看堆栈信息中线程的 CPU 使用率



10.2.4 堆分析工具 MAT

MAT 是 Memory Analyzer Tool（内存分析器）的简称，可以分析 HeapDump 的离线数据。下面简单介绍一下 Eclipse Memory Analyzer tool 版本，它可以从官网下载，网址为：<http://www.eclipse.org/mat/downloads.php>。安装后打开如图 10-32 所示的主界面。

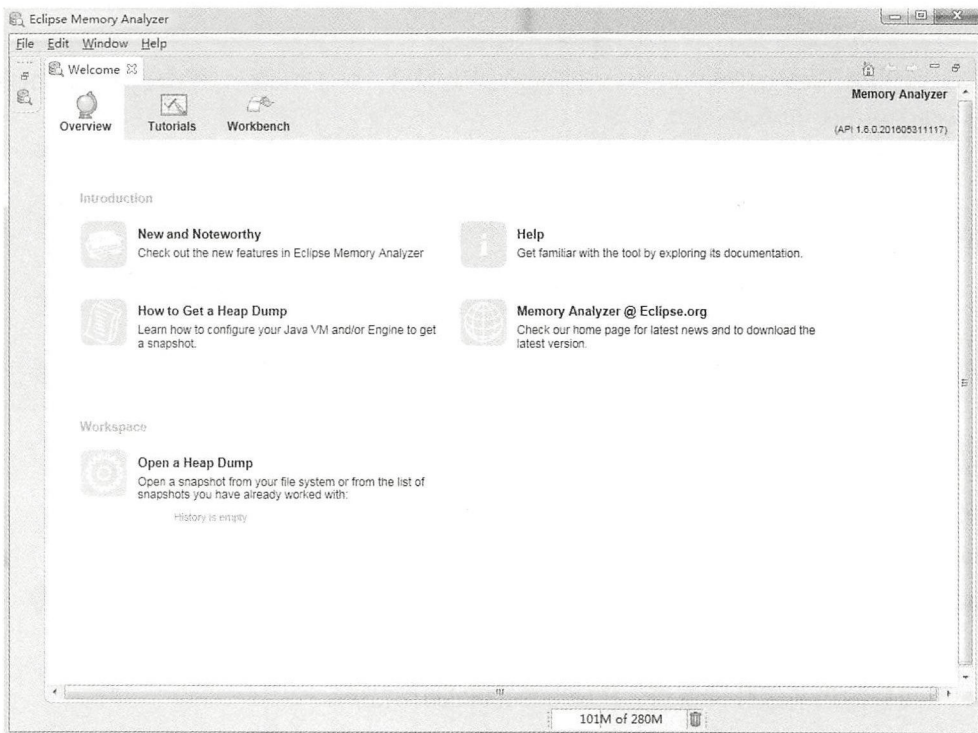


图 10-32 MAT 界面

MAT 分析的离线文件格式是 hprof 格式，可以使用 jmap 命令产生，也可以在 Java 程序的运行参数中添加以下参数来产生，在 Java 运行前设置才生效。

- `-XX:+HeapDumpOnOutOfMemoryError`: 打开 HeapDump 选项，在内存溢出时开始生产堆快照。
- `-XX:HeapDumpPath=d:/test.hprof`: hprof 文件保存的路径。

在主界面的下方“Workspace”区域，选择“Open a Heap Dump”选项，可以打开 dump 文件，MAT 会开始分析 dump 文件，然后罗列出可能存在的内存问题，如图 10-33 所示。



MAT 分析后默认打开“Leak Suspects”界面，展示了内存泄漏的可疑点，其中“Overview”区域显示了几个可疑内存问题所占用的内存大小，“Problem Supect X”区域显示了具体的内存问题，如占用多少内存、由什么原因造成等。

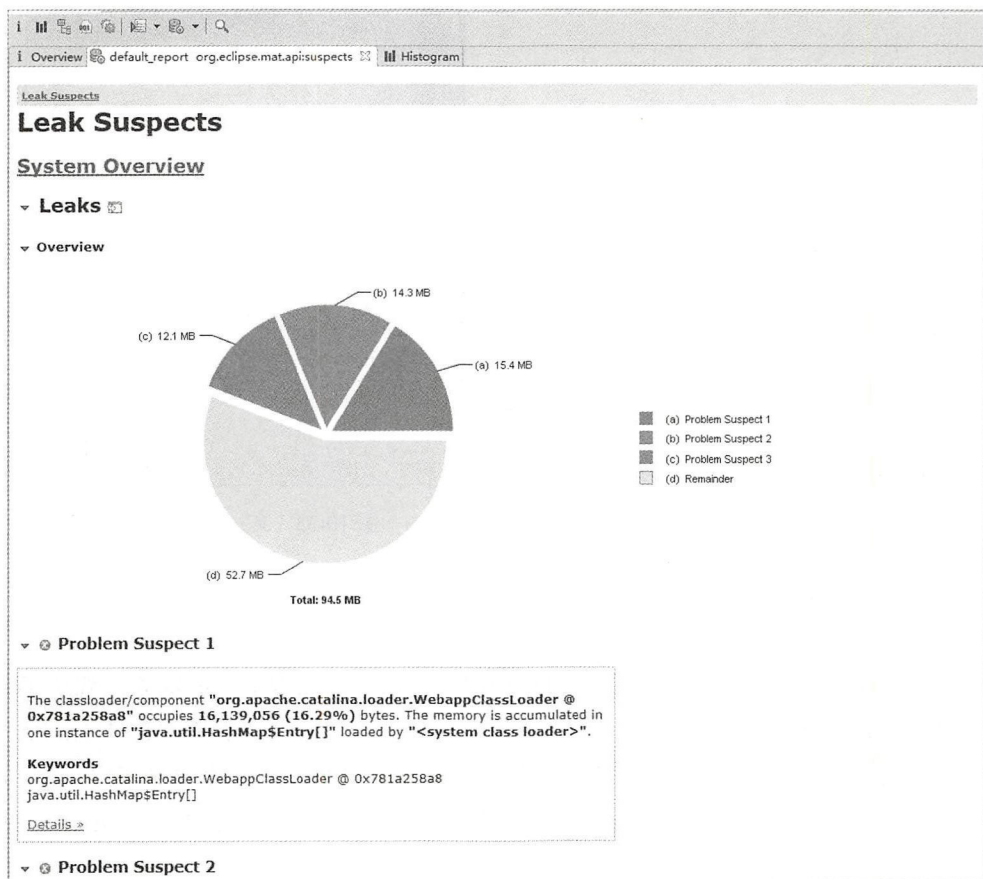


图 10-33 内存溢出问题

10.2.5 JConsole

JConsole 是 JDK 自带的比较全面的 Java 监控分析工具，使用起来简单方便。可以从命令行或 GUI 界面运行。类似的还有 VisualVM，这里就不一一介绍了。理解了 JConsole 的监控方式和原理，其他工具也是类似的。下面介绍使用 JConsole 进行监控的步骤。

(1) 获得 Tomcat 的 JMX 端口号“jmx.port=6914”，如图 10-34 所示。



(2) 获取 Tomcat 启动脚本配置信息后，在命令行中输入“jconsole”或双击 jconsole.exe，打开 JConsole，显示界面如图 10-35 所示。在“JConsole：新建连接”对话框中，输入被监控的目标服务器的 IP 地址和端口号。

```
# Path of application log
app.log.path=/opt/log
app.path.prefix=/nfs/war/tomcat/
shutdown.port=10014
ajp.port=8024
http.port=8094
https.port=8054
jmx.port=6914
jvmRouteName=tomcatServer215-14
```

图 10-34 获得 JMX 端口

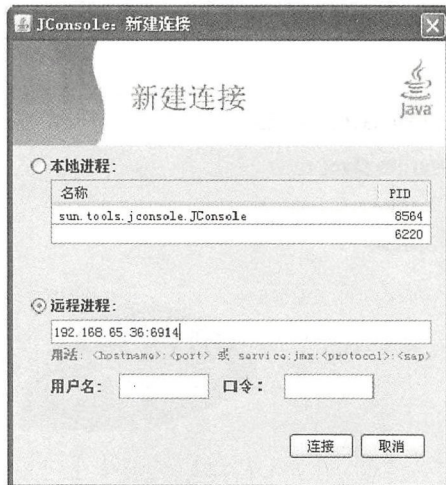


图 10-35 JConsole 新建连接

(3) 单击“连接”按钮后，打开“Java 监视和管理控制台”窗口，可以看见 4 个图，分别是堆内存、线程、类和 CPU 使用情况，如图 10-36 所示。

(4) 在“Java 监视和管理控制台”窗口中单击“内存”选项卡，可以放大观察 JVM 的堆内存使用情况，如图 10-37 所示。在“图表”下拉框中选择“内存池 ‘PS Eden Space’”选项，显示为新生代的垃圾回收状况图。在实例中新生代的垃圾回收状况正常，建议运行一段时间后，也注意观察一下老年代的垃圾回收情况，为此在“图表”下拉框中选择“内存池 ‘PS Old Gen’”选项，如果每次垃圾回收后，“PS Old Gen”波谷都有升高，说明可能存在一定的内存泄露，有些对象不能被 FGC 回收。还有一个标准，一般 FGC 以后，老年代的内存堆使用率会减少到 50%左右，最大为 65%，超过这个阈值，就应该查看跟踪代码了。

(5) 在“Java 监视和管理控制台”窗口中单击“线程”选项卡，可以查看线程阻塞和死锁情况，如图 10-38 所示。单击“检测到死锁”按钮可以检测 Java 应用死锁情况。

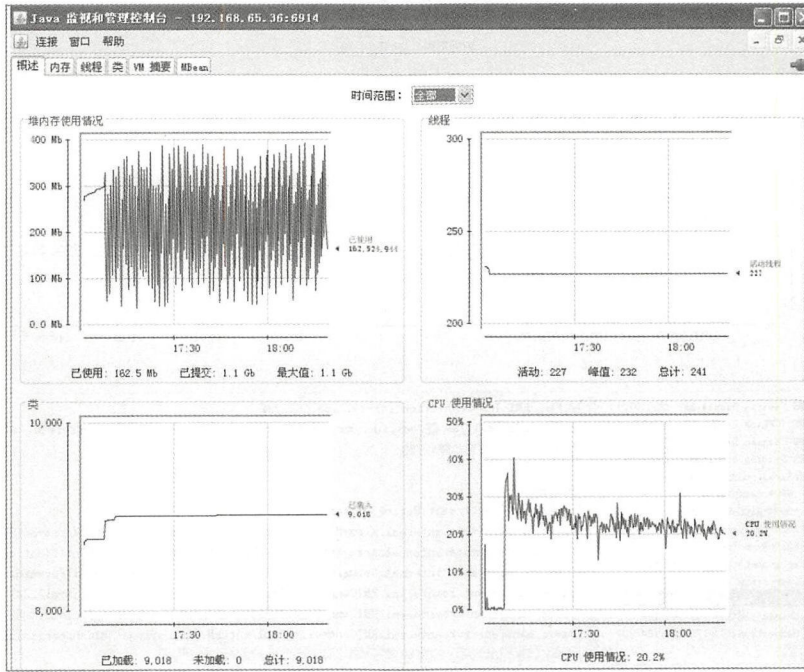


图 10-36 “Java 监视和管理控制台”窗口

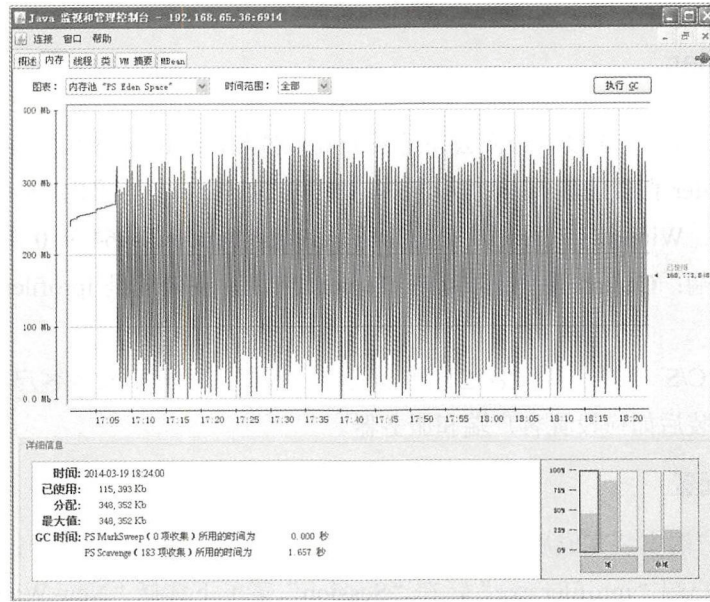


图 10-37 内存图

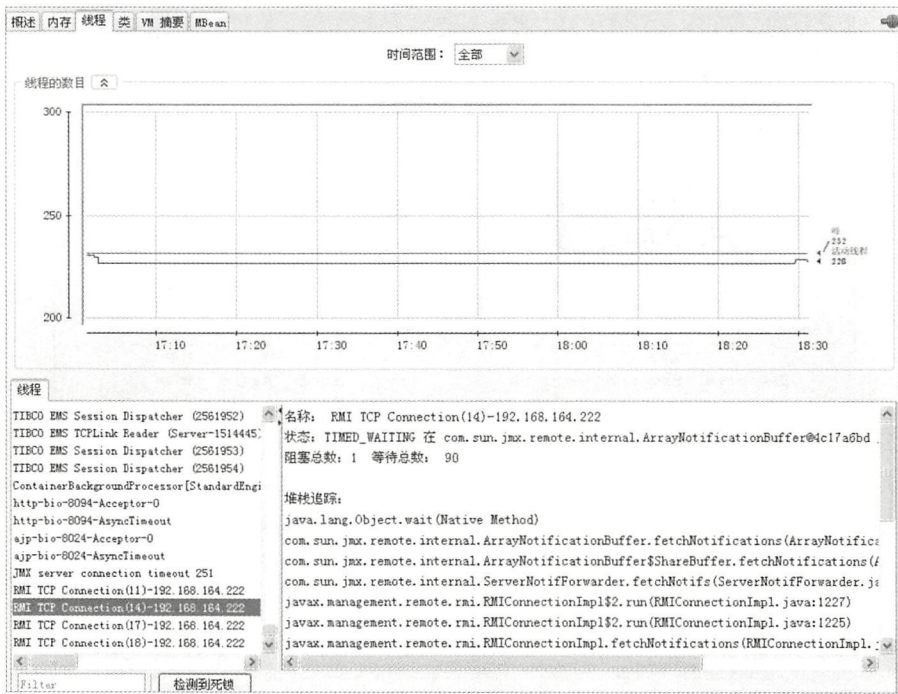


图 10-38 线程图

10.2.6 JProfiler

JProfiler 是一个由 EJ 技术有限公司开发并商业授权的 Java 剖析工具，可以剖析 JVM 内存情况。JProfiler 直观的界面有助于更快解决性能瓶颈。本书实例中使用的版本如下。

- 客户端：Windows 7 系统，安装的是 jprofiler_windows-x64_9_0_3 软件。
- 服务器端：Centos 系统上安装有 Tomcat 应用，安装的是 jprofiler_linux_9_0_3.sh 文件。

JProfiler 是 C/S 模式，其安装过程比较简单，根据提示分别安装客户端和服务端即可。下面讲解安装后如何设置客户端和服务端。

1. 客户端配置

JProfiler 客户端安装完成后，需要对客户端进行配置，具体步骤如下：

- (1) 打开客户端“jprofiler.exe”，在“Session”菜单下选择“New Window”选项，弹

出 “Quick start” 对话框，选择 “Profile an application server, locally or remotely” 选项。

(2) 单击 “下一步” 按钮，在 “Integration Wizard” 界面中，选择应用服务器版本如 “Tomcat7”。

(3) 单击 “下一步” 按钮，在 “Local or Remote” 界面中，选择 “on remote computer”，同时选择操作系统相关信息，如 “linux x86, amd64”。

(4) 单击 “下一步” 按钮，在 “Profiled JVM” 界面中，选择 JDK 的版本，如 “sun”、“1.7.0”（对应服务器端 jdk 版本）或 “hotspot”。

(5) 单击 “下一步” 按钮，在 “Startup Mode” 界面中，选择第二个选项 “startup immediately. connect later with jprofiler GUI”，这表示 JProfiler 启动会在 JProfiler 界面中设置连接。

(6) 单击 “下一步” 按钮，输入要监控连接的远程服务器的 IP 地址。

(7) 单击 “下一步” 按钮，输入 JProfiler 服务器端的安装位置，如 “/usr/local/jprofiler9”。

(8) 将服务器上 Tomcat 的 bin 目录下的 startup.sh 下载到客户端所在的计算机上。

(9) 单击 “下一步” 按钮，选择下载的 startup.sh 文件的存放路径。

(10) 单击 “下一步” 按钮，选择上一步中下载的 startup.sh 脚本；再单击 “下一步” 按钮，设定一个端口，默认是 8849；之后生成一个 startup_jprofiler.sh 脚本，而这个脚本就比原来的 startup.sh 多了如下一段内容，其中添加了 JProfiler 服务器端的启动路径和启动脚本。

```
# The following lines have been added by the
# application server integration wizard of JProfiler
CATALINA_OPTS="-agentpath:/usr/local/jprofiler9/bin/linux-x86/libjprofilerti.so=port=8849,nowait$
CATALINA_OPTS"
export CATALINA_OPTS
# end of modifications
```

此时，JProfiler 客户端会给出如下的一段文本提示，用来帮助配置 JProfiler 的服务器端。

```
Integration type: [Generic application]
Selected JVM: Sun 1.7.0 (hotspot)
Startup mode: Wait for JProfiler GUI
```

(1) Please insert

```
-agentlib:jprofilerti=port=8849 -Xbootclasspath/a:/usr/local/jprofiler9/bin/agent.jar
```



```
into the start command of your remote application right after the java command.
```

(2) Please add

```
/usr/local/jprofiler9/bin/linux-x64
```

```
to the environment variable LD_LIBRARY_PATH.
```

```
A remote session named Remote application on ***.***.***.
```

```
** will be created that connects to a running instance of the remote application ...
```

(11) 将上一步生成的 startup_jprofiler.sh 脚本上传到 Tomcat 的 bin 目录下，并赋予执行权限。

```
chmod 777 startup_jprofiler.sh
```

至此，客户端配置完成，接下来开始配置服务器端。

2. 服务器端配置

(1) 配置环境变量。

在 JProfiler 服务器端安装的系统上，修改环境配置文件 “/etc/profile”，在文件的末尾添加如下语句。如果是 32 位的操作系统，则把目录 linux-x64 换成 linux-x86 即可。

```
JPROFILER_HOME=/usr/local/jprofiler9/bin/linux-x64
```

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JPROFILER_HOME
```

执行完如下命令，系统的环境配置将生效。

```
source /etc/profile
```

(2) 编辑 Tomcat 的启动文件 “catalina.sh”。

使用如下命令，开始编辑 Tomcat 的启动文件。

```
vi /usr/local/tomcat/bin/catalina.sh
```

添加 “-agentlib:jprofilerti=port=8849 -Xbootclasspath/a:/usr/local/jprofiler9/bin/agent.jar” 内容到 CATALINA_OPTS 中，文件片断如下所示。

```
CATALINA_OPTS="$CATALINA_OPTS -Xms128m -Xmx128m $JPDA_OPTS  
-agentlib:jprofilerti=port=8849 -Xbootclasspath/a:/usr/local/jprofiler9/bin/agent.jar "
```

(3) 配置防火墙。CentOS 可以通过 iptables 限制 8849 端口仅供 JProfiler 客户端访问。

```
JProfiler> Protocol version 23
JProfiler> Using JVMTI
JProfiler> 64-bit library
JProfiler> Listening on port: 8849.
JProfiler> Native library initialized
JProfiler> Waiting for a connection from the JProfiler GUI
```

3. 使用 JProfiler

首先启动 JProfiler 客户端软件,再选择 JProfiler 界面中的菜单“Session”→“Start Center”→“Open Session”,在打开的对话框中,Available session configurations 列出了刚才配置的连接,选中就可以开始使用了,如图 10-39 所示。

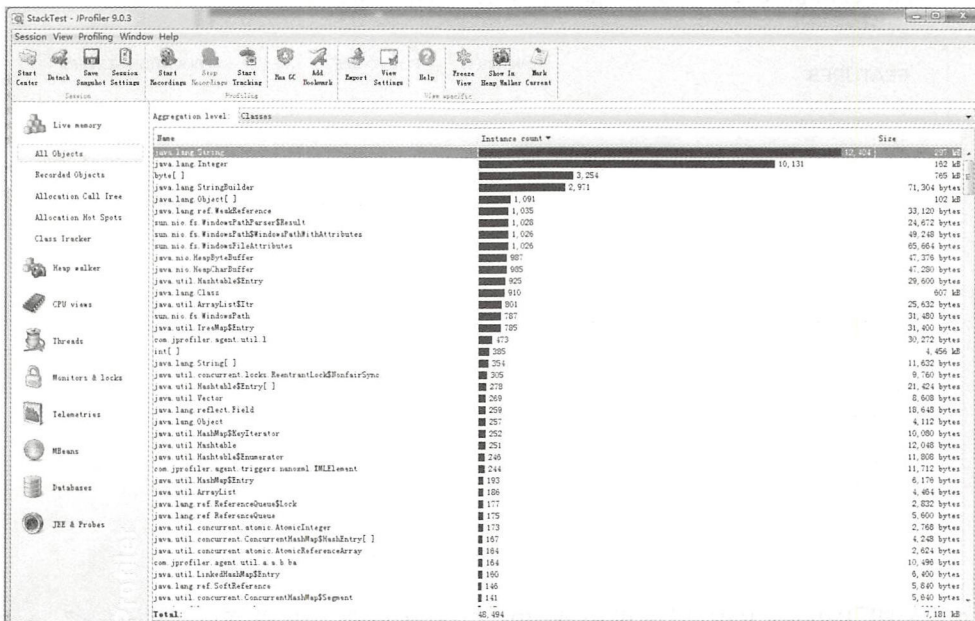


图 10-39 JProfiler 主界面

服务器端 Tomcat 的 \$CATALINA_HOME/logs/catalina.out 日志文件会显示 JProfiler 连接和中断时的信息。当连接 JProfiler 时，日志会显示以下的 JProfiler 连接信息。

质量全面管控：从项目管理到容灾测试

```
JProfiler> Using dynamic instrumentation
JProfiler> Time measurement: elapsed time
JProfiler> CPU profiling enabled
JProfiler> Hotspot compiler enabled
JProfiler> Starting org.apache.catalina/startup/Bootstrap
```

当中断 JProfiler 时，日志会显示以下的 JProfiler 断开信息。

```
JProfiler> Disconnected. Waiting for reconnection.
JProfiler> Listening on port: 8849.
```

4. JProfiler 与 Eclipse IDE 整合

在本地调试 Java 程序时，有时希望直接在 Eclipse 中快速调出 JProfiler 程序，以便更加快捷地跟踪 JVM 情况。下面讲述如何整合 Eclipse IDE 与 JProfiler 客户端。

(1)到 JProfiler官网查看支持的 Eclipse IDE 版本,如图 10-40 所示,网址为:<https://www.ej-technologies.com/products/jprofiler/java-profiler-IDE.html>。

FEATURES	OVERVIEW	PLATFORMS	IDES	APPLICATION SERVERS
JProfiler can be integrated into the following IDEs:				
IDE	Windows	Linux	Unix	OS X
eclipse 3.x, 4.x SEE MORE >	✓	✓	✓	✓
IntelliJ IDEA 9.x, 10.x, 11.x, 12.x, 13.x, 14.x, 15.x * SEE MORE >	✓	✓	✓	✓
NetBeans IDE > 6.x, 7.x, 8.x SEE MORE >	✓	✓	✓	✓
Oracle JDeveloper 10.1.3, 11g SEE MORE >	✓	✓	✓	✓

图 10-40 JProfiler 整合的 IDE 版本

(2) 关闭 Eclipse IDE 后，再启动 JProfiler 客户端；选择菜单“Session”→“IDE Intergrations”选项，如图 10-41 所示；选择 Eclipse 版本，如“Eclipse 4.3”，再单击“Integrate”按钮，然后选择 Eclipse 所在的安装目录，如“D:\Java\eclipse”。

(3) 整合成功后，显示如图 10-42 所示的成功信息。

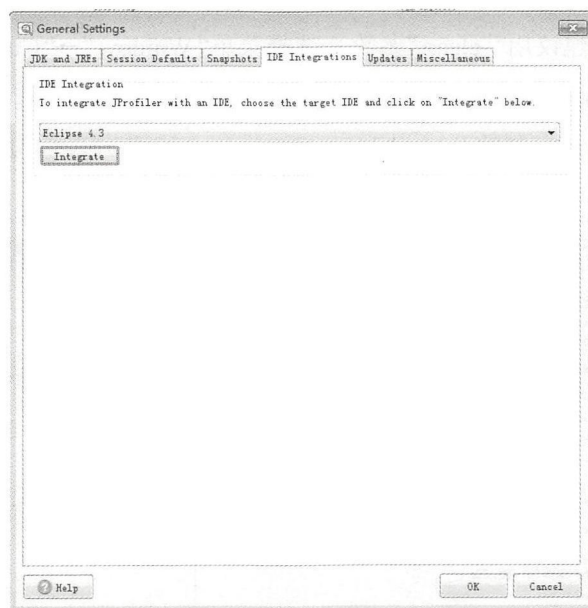


图 10-41 JProfiler 整合 IDE

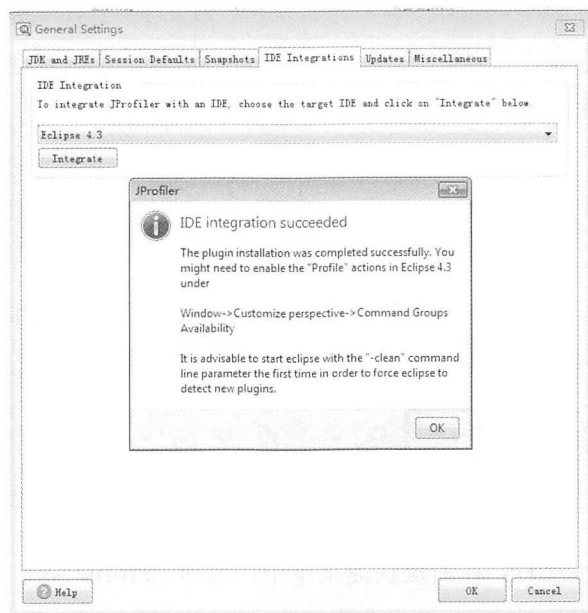


图 10-42 IDE 整合成功信息

质量全面管控：从项目管理到容灾测试

(4) 打开 Eclipse (建议用“-clean”参数), 选择菜单“Window”→“Customize Perspective”, 打开“Customize Perspective”对话框, 如图 10-43 所示。在对话框中选择“Command Groups Availability”选项卡, 在左侧的“Available command groups”列表中勾选“Profile”复选框, 单击“OK”按钮即可。

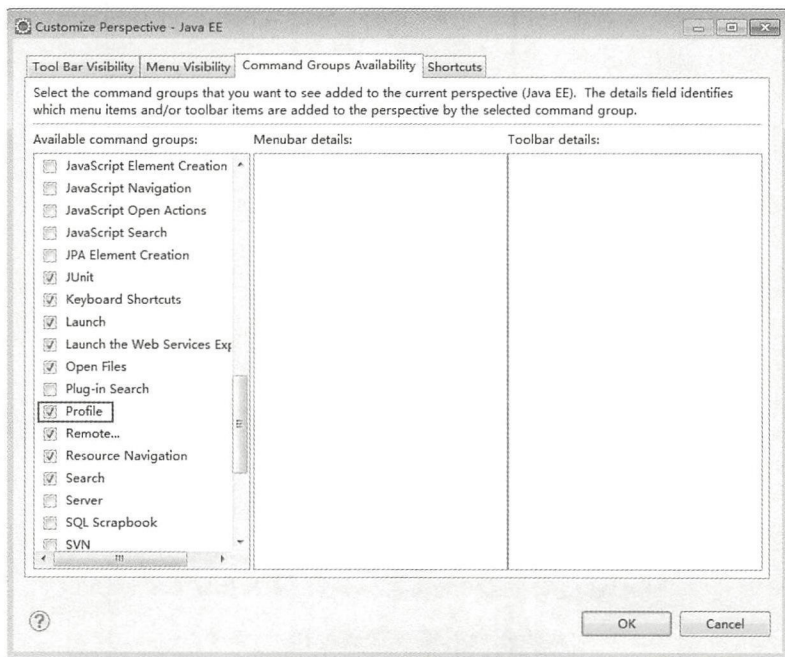


图 10-43 勾选 Profile 复选框

(5) 配置完成后, 在 Eclipse 的菜单栏上可以看到增加了“Profile”按钮, 如图 10-44 所示。



图 10-44 JProfiler 按钮

(6) 选择一个 Java 工程, 在右键快捷菜单中, 选择“Profile As”→“Java Application”选项, 如图 10-45 所示。

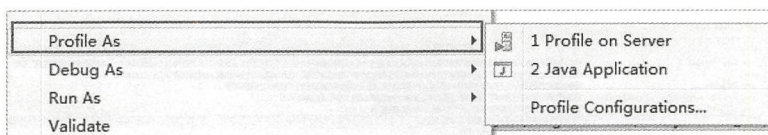


图 10-45 Profile As 选项

(7) JProfiler 会自动打开，如图 10-46 所示。

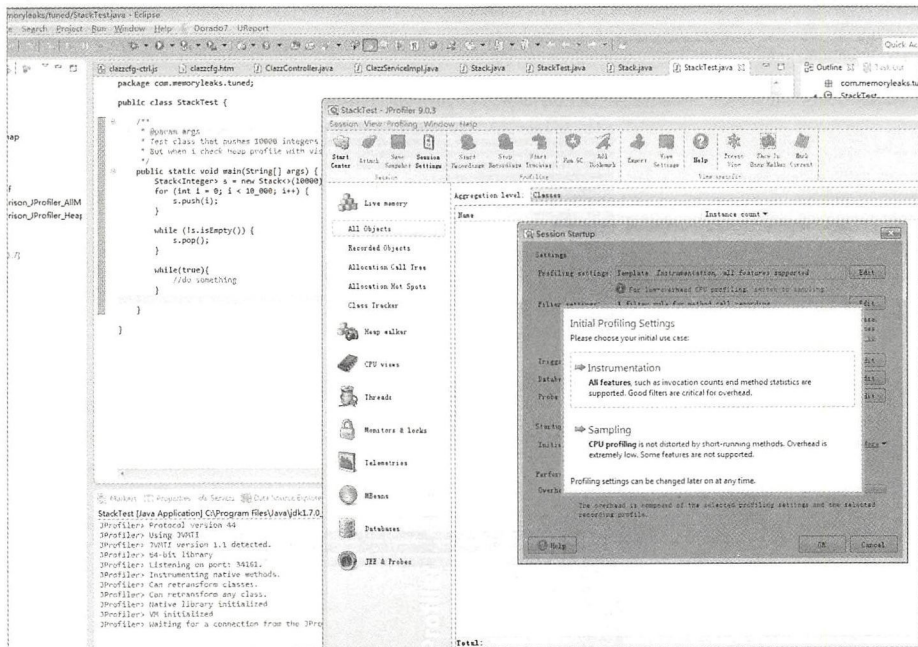


图 10-46 Profiler 从 IDE 中打开

5. 实例

当使用 JProfiler 监控一个 Java 应用程序时，在 Live summary 区域下的“**All Objects**”中发现 Integer[] 对象一直在增长，没有回收，如图 10-47 所示。而调优后，可以看到 Integer 对象已经不在前十名的对象列表上。

也可以在 Heap walker 中查看对象数量，如图 10-48 所示。调优前可以看到有 10153 个 Integer 对象，而调优后，在程序运行差不多时间之后，Integer 对象的数量已经难以在列表上看到，这表明优化有一定的成效。

质量全面管控：从项目管理到容灾测试

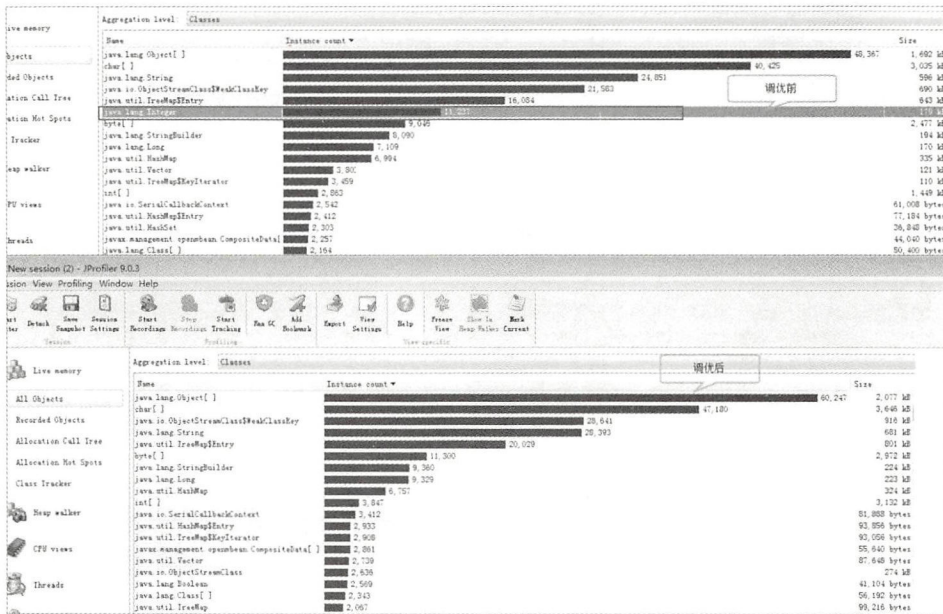


图 10-47 All Objects 调优前后对比

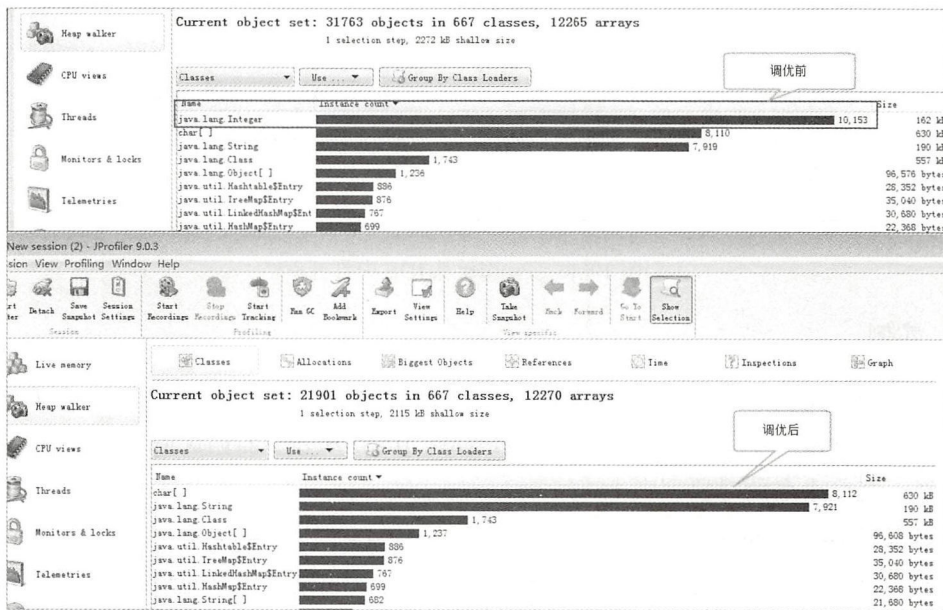


图 10-48 Heap walker 调优前后对比

10.3 数据库性能分析

基于以往经验，至少有 80% 的应用程序在优化数据库后性能有明显的提升。在监控数据库的时候，首先是观察数据库服务器的硬件资源健康状况，如 CPU、Memory、Disk、网络等，看看 CPU 使用率是否过高，内存换页率是否过于频繁，磁盘延迟时间是否过长，网络丢包率是否过多等。

其次在数据库硬件运行性能正常时，才应该开始查看哪个应用在访问数据库。确定访问同一数据库的所有服务的响应时间慢，还是单个服务的性能有问题。如果单个服务的数据库响应时间很慢，那么应该深入研究该服务与数据库之间的通信状况，如正在执行哪些查询语句，每个请求执行频率有多高，每次查询返回多少行的记录等。

最后，即使数据库查询方式是正常的，但数据库还是性能低下，那就应该查询数据库的最大连接数和当前连接数，是否连接池设置过小而无法提供足够的连接。

在压力测试执行结束后，可以查看 AWR 报告。如果是 MySQL，没有 AWR 报告，可以分析 Performance_schema 性能检测库和 MySQL 的慢日志（slow log）。

10.3.1 软解析和硬解析

SQL 语言共分为 4 大类：数据查询语言 DQL、数据操纵语言 DML、数据定义语言 DDL 和数据控制语言 DCL。

（1）数据查询语言 DQL 的基本结构是由 SELECT 子句、FROM 子句、WHERE 子句组成的查询块。

（2）数据操纵语言 DML 语句用于操作数据库对象中包含的数据，操作的单位是记录，包括数据库的增（INSERT）、删（UPDATE）、改（DELETE）操作。

- Insert 语句：向数据表插入一条记录。
- Delete 语句：删除数据表中的一条或多条记录，也可以删除数据表中的所有记录，但它的操作对象仍是记录。
- Update 语句：用于修改已经存在表中记录的内容。

（3）数据定义语言 DDL 语句用于操作对象和对象的属性，对象包括数据库本身和数据



库对象如表、视图等。DDL 语句对这些对象和属性的管理定义具体表现为 CREATE、DROP、ALTER、TRUNCATE。

- Create 语句：可以创建数据库和数据库的对象。
- Drop 语句：可以删除数据表、索引、触发程序、条件约束和数据表的权限等。
- Alter 语句：修改数据表定义和属性。
- Truncate 语句：删除数据表，但不能用于参与了索引视图的表。

(4) 数据控制语言 DCL 用来授予或回收访问数据库的某种特权，控制数据库操纵事务发生的时间和效果，以及对数据库实行监视等。

- GRANT：授权。
- ROLLBACK [WORK] TO [SAVEPOINT]：回退到某一点。
- COMMIT [WORK]：提交。

在这些 SQL 语句中，DDL 和 DCL 只会执行硬解析，DQL 和 DML 根据情况选择执行硬解析或软解析。SQL 语句执行时将进行以下几个步骤的处理过程来确定如何解析 SQL：

(1) 语法检查 (syntax check)。

检查 SQL 语句的拼写是否有语法错误。

(2) 语义检查 (semantic check)。

检查 SQL 语句中的访问对象是否存在和该用户是否具备相应的权限。

(3) 共享池检查 (Shared Pool check)。

检查 SQL 语句是否在共享池中存在，如果没有则加载 SQL 语句。

(4) 对 SQL 语句进行解析 (parse)。

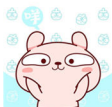
利用内部算法对 SQL 语句进行解析，生成解析树 (parse tree) 及执行计划 (execution plan)。

(5) 执行 SQL，返回结果 (execute and return)。

第一个步骤决定软解析或硬解析，软解析跳过第一步，执行后面几步，而硬解析需执行所有步骤。

- 软解析 (Soft Parse)

数据库利用内部的哈希算法来取得该 SQL 语句的 Hash 值，然后在共享池里查找是否存在该 Hash 值以及版本是否一致。假设存在，则将此 SQL 语句与 cache 中的语句进行比较。若“相同”，就利用已有的解析树和执行计划，而省略了优化器的相关工作，这就是软



解析的过程。

● 硬解析 (Hard Parse)

如果软解析查找与比较步骤中有一个不成立,那么优化器都将创建解析树、生成执行计划。这个过程就是硬解析。创建解析树、生成执行计划对于 SQL 语句的执行来说是开销昂贵的动作,所以开发人员应尽量保持代码一致,多绑定参数,减少硬解析。使用 cursor_sharing 参数应权衡利弊,需要考虑使用 similar 与 force 带来的影响。

10.3.2 SQL 执行计划分析

什么是 SQL 执行计划? SQL 执行计划是一条 SQL 查询语句的执行过程或访问路径的描述。SQL 执行计划是一个需要深入研究的课题,下面通过对一个实例的分析简单了解一下。这里可以使用最常见的工具,如 PL/SQLDeveloper 访问 Oracle 数据库。

PL/SQLDeveloper 工具允许查看一个图形化的执行计划,按 Ctrl+E 快捷键打开执行计划界面。查看一个执行计划时,SQL 查询语句并没有实际运行,其实际返回的行数和数据只能在运行完后才能得到。在本实例中,如图 10-49 所示,由于没有将建索引的字段作为查询条件,结果将导致对全表进行扫描,其占用 CPU 的使用率达到 89%。索引缺失导致的全表扫描会造成数据增加时查询速度越来越慢,因此需要对 SQL 语句添加索引等优化了。

1. SQL 单次耗时 1.2 秒, 全表扫描

```
select count(*) as col_0_0_ from T_TMC_AT_DATA_REQUEST tmcdata0_ where (tmcdata0_ STATUS in ('1', '2'))
and tmcdata0_ submit_Flag=:3 and tmcdata0_ EXT_TXN_ID=:4 and tmcdata0_ TMC_ORG_PARTY_ID=:5
```

Description	Objectowner	Objectname	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			22896	1	26
SORT AGGREGATE				1	26
PARTITION RANGE ALL			22896	1	26
TABLE ACCESS FULL	CPS	T_TMC_AT_DATA_REQUEST	22896	1	26

2. SQL 单次耗时 0.36 秒, 全表扫描

```
SELECT count(*) FROM T_TMC_AT_DATA_REQUEST t1, T_TMC_INF_AIR_DATA t2 where t1.ID_TMC_AT_DATA_REQUEST = t2.ID_TMC_TXN
and t1.tmc_org = :1 and t1.txn_type = :2 and t2.ticket_no = :3 and t1.submit_Flag = '1' and t1.status in ('0', '1')
```

Description	Objectowner	Objectname	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			5435	1	37
SORT AGGREGATE				1	37
NESTED LOOPS			5435	1	37
NESTED LOOPS			5433	1	20
PARTITION RANGE ALL			5433	1	20
TABLE ACCESS FULL	CPS	T_TMC_INF_AIR_DATA	5433	1	20
INDEX UNIQUE SCAN	CPS	PK_T_TMC_AT_DATA_REQUEST	1	1	
TABLE ACCESS BY GLOBAL INDEX ROWID	CPS	T_TMC_AT_DATA_REQUEST	2	1	17

Buffer Gets	Executions	Gets per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text
2,004,014,000	29,653	90,514.00	75.65	35,470.72	95.3	42.7	9vz7y2tkr16ou		select count(*) as col_0_0_ fr...
650,603,417	29,651	21,942.04	18.34	10,731.47	89	32.4	ffcnd88xwqad		SELECT count(*) FROM T_TMC_AT...
76,581,676	598	128,063.00	2.16	1,030.69	89.4	38.9	dy28998x1sq7w		select * from (select tmcnfs...

图 10-49 全表扫描

10.3.3 数据库连接数监控

Tomcat 连接数据库都是使用 JDBC 连接池，在高并发时，连接池的数量可能不够用，所以需要监控连接池的数量，可以在服务器端使用 netstat 命令来实现。

(1) 方法一：在 Tomcat 服务器中输入如图 10-50 所示的命令，可以查看 Tomcat 连接 Oracle 数据库连接池的情况，并且还可以排序。其中“1530”是数据库的端口，“uniq-c”会统计唯一行的数量。

```
[root@tomcat063 10313]# netstat -an | grep 1530 | awk '{print $(NF-1)}'|sort | uniq -c
2 ::ffff:192.168.6.146:1530
```

图 10-50 DB 连接池监控命令

(2) 方法二：直接使用“grep-c”命令统计数据库端口的连接数，而不是“uniq-c”命令。

```
netstat -anp | grep -c {数据库端口|数据库 IP}
```

命令如下所示，其中“grep -c 1530”匹配含有端口号 1530 的行，并统计总行数。

```
netstat -anp | grep -c 1530
```

另外，在数据库上可以使用以下 SQL 语句，查看数据库的当前连接数、最大连接数和并发连接数。

(1) 查看当前数据库的参数设置，SQL 语句如下：

```
show parameter service_names;
```

(2) 查看当前的连接数，SQL 语句如下：

```
select count(*) from v$process;
```

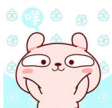
(3) 查看数据库允许的最大连接数，SQL 语句如下：

```
select value from v$parameter where name = 'processes';
```

(4) 查看并发连接数，SQL 语句如下：

```
select count(*) from v$session where status='ACTIVE';
```

(5) 查看当前各个客户端计算机的连接数，SQL 语句如下：




```
select count(*), machine from v$session group by machine;
```

(6) 查看当前客户端计算机名为 RHEL 的所有 session 数, SQL 语句如下:

```
select count(*) from v$session where machine = 'RHEL';
```

(7) 查看某台服务器的各个 schema 连接情况, SQL 语句如下:

```
select m.MACHINE,username,count(*) from v$session m Where MACHINE='inf-qz01' group by  
m.MACHINE,username
```

(8) 查看所有服务器连接到某个 schema 的连接数情况, SQL 语句如下:

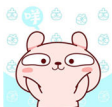
```
select m.MACHINE,username,count(*) from v$session m Where username='DDP' group by  
m.MACHINE,username
```

10.3.4 Oracle 数据库性能诊断报告 AWR

1. AWR 简介

在 Oracle 10g 之前, Oracle 使用 “statpack” 作为性能分析工具, 这个工具是 TXT 格式的 report, 阅读起来比较麻烦。而且 Oracle 用户的连接将产生会话, 当前的会话记录被保存在 v\$session 视图中, 处于等待状态的会话会被复制一份放在 v\$session_wait 视图中, 当该连接断开后, 原来在 v\$session 视图和 v\$session_wait 视图中的连接信息就会被删除。Oracle 10g 及之后的版本保留了 v\$session_wait 视图中的信息, 并增加了 v\$active_session_history(ASH)视图, 记录每个活动会话在 v\$session_wait 中最近 10 次的等待事件。ASH 的采样数据保存在内存中, 但分配给 ASH 的内存空间是有限的, 当所分配的空间占满后, 旧的记录就会被覆盖, 而且数据库重启后, 所有的这些 ASH 信息都会消失。因此, 长期检测 Oracle 的性能是不可能的。

在 Oracle 10g 中提供了永久保留 ASH 信息的方法, 这就是 AWR (Automatic Workload Repository)。全部保存 ASH 信息是非常耗费时间和空间的, AWR 采用的策略是每小时对 v\$active_session_history 视图采样一次, 并将信息保存到磁盘中, 保留 7 天, 7 天后旧的记录才会被覆盖。这些采样信息被保存在视图 wrh\$_active_session_history 中, 而这个采样频率 (1 小时) 和保留时间 (7 天) 是可以根据实际情况进行调整的, 这就给测试人员提供了更加有效的系统监测。通过解读 AWR 就能知道 Oracle 有哪些性能问题, 以及将来可能出



现什么问题，将 AWR 与 Oracle EM 管理工具结合能够使测试人员更轻松地掌握 Oracle 数据库的性能动态。

2. 生成 AWR 报告

生成 AWR 报告的方法有很多，下面只介绍采用 Toad 工具生成 AWR 报告的方法。在压力测试前执行命令“`exec dbms_workload_repository.create_snapshot;`”，性能测试结束后再次执行相同命令，这样就会采集到测试前后的会话记录快照 snapshot，如图 10-51 所示。通过查询 `DBA_HIST_SNAPSHOT` 视图，可获得系统中产生的快照记录。

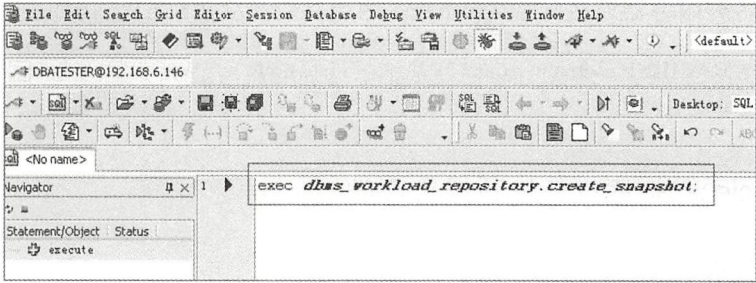


图 10-51 Toad 生成 AWR 报告

在 Toad 的菜单栏中选择“Database”→“Monitor”→“ADDM&AWR Reports”选项，然后在打开的设置界面中（如图 10-52 所示），选择性能测试期间的两次快照并单击“生成报告”按钮（高亮的绿色按钮），这样就会生成 AWR 报告。界面中的“Starting Snapshot”是报告收集时间段内开始时间的快照 ID，“Ending Snapshot”是结束时间的快照 ID。

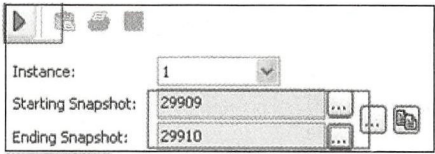
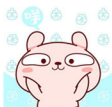


图 10-52 单击“生成报告”按钮生成 AWR

3. 解读 AWR 报告

生成 AWR 后，如何正确解析 AWR，并发现其中的性能问题就变得至关重要了。下面介绍 AWR 报告中包括的几项内容。

(1) 报表头信息是数据库实例的相关信息，包括数据库名称、ID、版本号和主机名等信息，如图 10-53 所示。



WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
BR01	1517016693	ebr01	1	21-10月-15 20:10	11.2.0.1.0	NO

Host Name	Platform	CPU(s)	Cores	Sockets	Memory (GB)
idc-db01	Linux x86 64-bit	32	16	4	125.70

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	741	22-10月-15 16:00:31	95	1.6
End Snap:	743	22-10月-15 18:00:32	72	1.6
Elapsed:		120.03 (mins)		
DB Time:		1,159.29 (mins)		

图 10-53 AWR 报表头信息

- DB Name: 数据库名字。
- DB Id: 数据库 ID。
- Elapsed: 采样时间段。
- DB Time: 用户操作花费的时间, 不包括 Oracle 后台进程消耗的时间。如果 DB Time 远小于 Elapsed Time, 说明数据库比较空闲。

(2) 实例负载档信息, 如图 10-54 所示。

Load Profile				
	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.3	0.0	0.00	0.00
DB CPU(s):	1.2	0.0	0.00	0.00
Redo size (bytes):	197,575.1	944.5		
Logical read (blocks):	36,982.4	176.8		
Block changes:	649.2	3.1		
Physical read (blocks):	0.4	0.0		
Physical write (blocks):	25.5	0.1		
Read IO requests:	0.4	0.0		
Write IO requests:	10.5	0.1		
Read IO (MB):	0.0	0.0		
Write IO (MB):	0.2	0.0		
Global Cache blocks received:	154.5	0.7		
Global Cache blocks served:	156.7	0.8		
User calls:	2,025.4	9.7		
Parses (SQL):	797.1	3.8		
Hard parses (SQL):	19.9	0.1		
SQL Work Area (MB):	2.8	0.0		
Logons:	0.1	0.0		
Executes (SQL):	998.0	4.8		
Rollbacks:	162.9	0.8		
Transactions:	209.2			

图 10-54 AWR Load Profile



实例加载中显示了数据库的负载概况，与基线数据比较时，如果每秒或每个事务的负载变化不大，说明数据库比较稳定。单个的 AWR 报告只能说明当前应用的负载情况，并不能判断正常与否。但是如果全部 parses 超过每秒 300、Hard parses 大于每秒 100、Logons 大于每秒 2 个，就意味着数据库可能有争用问题。实例负载当中的各选项说明如下。

- Redosize: 每秒产生的日志大小（单位字节），可表示数据变更频率、数据库任务的多少。如果日志产生大小过于庞大，但是业务没有这类需求，那么应该关心一下哪些 DML 语句执行次数特别多，是否有比往常多的返回记录。
- Logical reads: 平均每秒数据库从内存读取的数据块（block）数量。计算公式为：Logical Reads=Consistent Gets + DB Block Gets。其中 Consistent Gets 是指在回滚段 Buffer 中的数据一致性所要读取的数据块总数，DB Block Gets 是指当前请求的数据块在 Buffer 中能够获得的总数。
- Blockchanges: 每秒修改数据块的数量。
- Physicalreads: 平均每秒数据库从磁盘读取的数据块数量。
- Physicalwrites: 平均每秒数据库写磁盘的数据块数量。
- Usercalls: 每秒用户调用次数，是数据库向应用服务器发起的登录、解析、执行、获取等操作。如果值比较大，有可能是过量切换上下文，最好看一下“SQL ordered by executions”中哪些 SQL 语句调用过多。
- Parses: 每秒解析次数，包括 fastparse、softparse 和 hardparse 三种解析数量的综合。fastparse 指的是直接在 PGA 中命中的情况（设置了 session_cached_cursors=n）；softparse 是指在 sharedpool（共享池）中命中的情形；hardparse 则是指都不命中的情况。软解析每秒超过 300 次意味着应用程序效率不高需调整，session_cursor_cache。
- Hardparses: 每秒产生的硬解析次数，若每秒超过 100 次，则说明可能绑定使用得不好，也可能是 sharedpool 设置不合理。
- SQL Work Area（MB）: Oracle 为了产生估算（workarea）处理的数据量。
- Logons: 每秒登录的次数。
- Executes: 每秒执行次数。
- Rollbacks: 每秒回滚次数。
- Transactions: 每秒产生的事务数，反映数据库任务繁重与否。



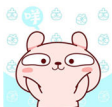
(3) 实例效率信息, 如图 10-55 所示。

Instance Efficiency Percentages (Target 100%)			
Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.95	In-memory Sort %:	100.00
Library Hit %:	99.92	Soft Parse %:	99.90
Execute to Parse %:	41.81	Latch Hit %:	99.22
Parse CPU to Parse Elapsed %:	101.47	% Non-Parse CPU:	98.91

图 10-55 实例效率

实例效率信息这部分的价值越接近 100 越好, 其中各选项的详细说明如下。

- **Buffer Nowait%:** 在缓冲区中获取 Buffer 的未等待比率。BufferNowait%值一般需要大于 99%, 否则可能存在争用, 可以在后面的等待事件中进一步确认。
- **Redo NoWait%:** 在 Redo 缓冲区中获取 Buffer 空间的未等待比率。当 RedoBuffer 达到 1MB 时, 就需要写到 redolog 文件中, 所以, 一般 RedoBuffer 设置超过 1MB 时, 不太可能存在等待 Buffer 空间分配的情况。RedoBuffer 设置为 2M, 对于内存总量来说, 应该不是一个太大的值。
- **BufferHit%:** 数据块在缓冲区中的命中率。通常应在 95%以上。若小于 95%, 则需要调整参数; 若小于 90%, 则可能是需要加 db_cache_size。一个高的命中率, 不一定代表这个系统的性能是最优的, 比如大量的非选择性的索引被频繁访问, 就会造成命中率很高的假相。但是一个比较低的命中率, 一般就会对这个系统的性能产生影响, 需要进行调整。命中率的突变往往是一个不好的信息, 如果命中率突然增加, 可以检查 topbuffergetSQL, 查看导致大量逻辑读的语句和索引; 如果命中率突然减小, 可以检查 topphysicalreadsSQL, 查看产生大量物理读的语句, 特别关注那些没有使用索引或者索引被删除的语句。
- **In-memorySort%:** 在内存中的排序率。如果低于 95%, 则可以通过适当调大初始化参数 PGA_AGGREGATE_TARGET 或者 SORT_AREA_SIZE 的值来解决。注意这两个参数设置作用的范围是不同的, SORT_AREA_SIZE 是针对每一个 session 设置的, 而 PGA_AGGREGATE_TARGET 则是针对所有 session 指定总的最大可以使用的 PGA 内存。
- **LibraryHit%:** SQL 语句在共享区的命中率。通常应该保持在 95%以上, 否则需要考虑优化, 如加大共享池、使用绑定变量或修改 cursor_sharing 等参数。



- **SoftParse%:** 软解析在总分析数的百分比。若小于 95%，需要考虑绑定；若小于 80%，那么就可以认为 SQL 语句基本没有被重用。
- **Execute to Parse %:** 一个语句执行和分析了多少次的度量。计算公式为：
- **Execute to Parse=100 * (1 -Parses/Executions)**。在本例中，差不多每执行 10 次需要解析 6 次。如果系统 Parses 的值大于 Executions，就可能出现该比率小于零的情况，通常说明共享池设置或者 SQL 语句效率存在问题。
- **LatchHit%:** Latch 锁命中率，Latch 是一种保护内存结构的锁。要确保该值大于 99%，否则存在严重的性能问题。当该值出现问题时，可以借助后面的等待时间和 Latch 分析问题。
- **ParseCPUtoParseElapsd%:** 解析需要的 CPU 使用率。计算公式为：
$$\text{ParseCPUtoParseElapsd}\% = 100 * (\text{parsetimecpu} / \text{parsetimeelapsed})$$
，即：解析实际 CPU 时间/（解析实际 CPU 时间+解析中等待资源时间）。如果该比率为 100%，则意味着 CPU 等待时间为零，没有任何等待。
- **% Non-Parse CPU:** 未解析的 CPU 比率。计算公式为：
$$\% \text{ Non-Parse CPU} = \text{round}(100 \times (1 - \text{PARSE_CPU} / \text{TOT_CPU}), 2)$$

其中 PARSE_CPU 是解析消耗的 CPU 时间，TOT_CPU 是总的 CPU 时间。如果这个值比较小，表示解析消耗的 CPU 时间过多。与 PARSE_CPU 相比，如果 TOT_CPU 很高，这个比值将接近 100%，说明计算机执行的大部分工作是 SQL 查询工作，而不是分析查询的工作。

(4) Cache（缓存）信息，如图 10-56 所示。

Cache Sizes				
	Begin	End		
Buffer Cache:	59,008M	59,008M	Std Block Size:	8K
Shared Pool Size:	5,888M	5,888M	Log Buffer:	91,416K

图 10-56 AWR 缓存信息

内存管理方式包括 MSMM、ASMM(sga_target)、AMM(memory_target)。Buffer Cache 和 Shared Pool Size 的 Begin 和 End 值在 ASMM、AMM 和 11gR2 MSMM 下会变化。如果 Shared Pool Size 一直收缩，则说明在收缩过程中一些 row cache 对象被锁住，可能导致前

台 row cache lock 的解析等待。如果 Shared Pool Size 一直在增长, 那说明 Shared Pool 原有的大小不足以满足需求, 有可能是大量的硬解析, 需结合解析信息和 SGA breakdown 做进一步分析。

本例中 Buffer Cache 和 Shared Pool Size 的 “Begin” 和 “End” 字段大小是一样的, 表明 Shared Pool 的大小够用。

(5) 共享池统计, 如图 10-57 所示。

Shared Pool Statistics		
	Begin	End
Memory Usage %:	75.12	75.13
% SQL with executions>1:	99.25	99.14
% Memory for SQL w/exec>1:	86.51	85.38

图 10-57 共享池统计

- **MemoryUsage%:** 正在使用的共享池的百分率。这个值应该长时间稳定在 75%~90%之间。如果这个值太低, 表明共享池设置过大, 带来额外的管理负担, 在某些条件下会导致性能的下降; 如果这个值太高, 会使共享池外部的组件老化, SQL 语句再次执行时可能会硬解析。
- **%SQLwithexecutions>1:** 这是在共享池中执行次数大于一次的 SQL 语句的统计百分比。系统连续运行相同的 SQL 语句, 这个数值才会接近 100%。实例中显示在共享池中几乎有 99%的 SQL 语句中运行次数多于一次, 仅剩的 1%可能已经在那里了, 只是系统没有去执行。
- **%MemoryforSQLw/exec>1:** 这是执行次数大于一次的 SQL 语句消耗的内存百分比。它在总体上与 %SQLwithexecutions>1 的值非常接近, 除非有某些查询任务消耗的内存没有规律。

(6) 等待 (wait) 事件, 如图 10-58 所示。

这一节显示了系统中最严重的 10 个等待, 按所占总等待时间的比例倒序。对于一个性能良好的系统来说, DB CPU 应该在 Top 10 的第一行, 否则说明系统大部分时间都用在等待上。本实例中, reliable message 是除了 CPU 事件最多的等待事件, 占用了 4%的 CPU 时间, 而 log file parallel write 是相对比较多的等待事件, 占用了 2.9%的 CPU 时间。

质量全面管控：从项目管理到容灾测试

Top 10 Foreground Events by Total Wait Time					
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		4327.7		90.1	
reliable message	294,051	192.1	1	4.0	Other
log file sync	161,950	139.3	1	2.9	Commit
gc current block 2-way	264,423	113.9	0	2.4	Cluster
gc cr block 2-way	243,647	111.3	0	2.3	Cluster
gc current grant busy	93,169	36.6	0	.8	Cluster
gc current block busy	13,307	28.1	2	.6	Cluster
gc cr block busy	12,197	25	2	.5	Cluster
enq: TX - row lock contention	2,175	18.9	9	.4	Application
SQL*Net message to client	4,667,330	6.3	0	.1	Network

图 10-58 Top10 等待事件

(7) SQL ordered by Elapsed Time, 如图 10-59 所示。

SQL ordered by Elapsed Time									
<ul style="list-style-type: none"> Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code. % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100 %Total - Elapsed Time as a percentage of Total DB time %CPU - CPU Time as a percentage of Elapsed Time %IO - User I/O Time as a percentage of Elapsed Time Captured SQL account for 50.7% of Total DB Time (s): 808 Captured PL/SQL account for 0.2% of Total DB Time (s): 808 									
Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text	
73.86	202	0.37	9.14	99.14	0.00	76uq18u079xu7	JDBC Thin Client select " as MultiSelectionFla...		
65.84	202	0.33	8.15	99.22	0.00	173y74bizb858	JDBC Thin Client select count(*) as cnt, sum(B...		
62.54	387	0.16	7.74	98.11	0.00	8tf5z1av6xhd7	JDBC Thin Client select count(1) as sCount from...		
51.35	181	0.28	6.36	92.24	0.00	02qgabu9kbbay	JDBC Thin Client select ft.SerialNo, ft.ObjectN...		
18.97	160	0.12	2.35	98.14	0.00	4fn3k0q0axk0s	JDBC Thin Client select fo.ObjectNo as ObjectNo...		
12.17	26,431	0.00	1.51	27.31	0.00	6w0tp7vrf7y8q	JDBC Thin Client update OBJECT_MAXSN set MaxSer...		
10.92	647	0.02	1.35	89.56	2.31	cductucahcfcx	JDBC Thin Client select CustomerID from CUSTOME...		
7.79	160	0.05	0.96	98.16	0.00	f300ci63ir168	JDBC Thin Client select count(*) as cnt from bu...		
7.28	743	0.01	0.90	83.72	0.00	a62d47war5dqb	JDBC Thin Client select FLOW_OBJECT.ObjectType ...		
6.64	26,595	0.00	0.82	24.32	0.00	1nmxq6nu0qzqd	JDBC Thin Client insert into CONTRACT_REPEAT_CH...		

图 10-59 SQL ordered by Elapsed Time

各列项在表格上方已有解释，在 AWR 报告这一节中需要关注 Elapsed Time 时间和 Executions 最多的 SQL 语句，可以看出哪些 SQL 语句占用 CPU 时间和执行次数最多。定位问题 SQL 后，可以单击 SQL Id 列的链接查看 SQL 语句的具体内容。

4. AWR 报告总结

以上只是介绍了 AWR 报告中的部分内容，通过 AWR 报告的解读能了解到数据库的性能变化，获知哪些 SQL 语句执行时间较长，以及如何去调优。当然这个过程需要大量的经验积累，AWR 的全部解读也是很难一下子解释的，这是一个可以深入研究的课题。

10.4 要点回顾

性能分析工具只是提供了一个参考，在实际的性能测试过程中，还需要结合实际和经验。本章了解了 JVM 的原理，介绍了垃圾回收机制以及如何利用 JVM 监控工具观察分析 JVM 堆使用情况。本章还讲述了系统监控工具 nmon 和常用的 Linux 命令 netstat、jps、top 等，介绍了 SQL 计划以及掌握 AWR 报告的收集方式和分析要点。

- 硬件资源监控相关内容：nmon、netstat、ps、top 和 pmap。
- JVM 监控和分析工具 MAT、JConsole 和 JProfiler。
- lsof、jstat、jmap 和 jstack 等命令的用法。
- SQL 执行计划分析对性能的影响。
- 解读 AWR 报告。



第 11 章

监控平台与故障排查

对于现在的互联网公司来说，一个产品的服务器就可能有成百上千个，其应用环境也各式各样，如独立服务器、集群服务器和云服务器等。无论是 Web 服务器、应用服务器还是数据库服务器，硬件状态如果出现异常都需要及时给出警报，并通知相关责任人，如运维人员、DBA 等。仅靠单纯的人工实时监控，满足不了这样的需求，所以监控平台也就应运而生了。优秀的监控平台可以捕获异常的事件，通过邮件、短信等方式在第一时间通知到责任人。

之前，某一个电商网站在“双 11”和“双 12”等购物节期间，交易额突破了 100 亿元。这么高的交易量一旦出现系统崩溃，损失是非常巨大的，而且会流失大量的客户和订单。哪怕出现一个交易错误，也需要应急操作，分析问题所在，升级服务器容量等配置，将可能出现的相关问题尽量规避掉。

那么监控系统运行时，需要关注什么？出现故障了该怎么办？本章将介绍监控系统与故障排查，主要包括如下内容：

- Zabbix 的架构和配置。
- Grafana 的配置和使用。
- 服务器故障排查的关键步骤。

11.1 监控系统

随着企业信息化的持续深入发展，企业系统更加多样化和复杂化，相对应的系统运维和监控变得更加重要。监控系统是整个运维环节乃至整个产品或项目中最重要的一环。



那么, 怎样才是好的监控平台呢? 首先, 在保障系统业务正常运行时, 平台能够主动监控服务器、网络设备、数据库和应用程序的运行状态, 能够及时发现隐患。其次, 在故障出现时, 能够智能化警告 (如使用邮件、微信、短信等), 能够快速获取必要的信息, 迅速定位问题症结, 并解决故障, 减少处理时间。再次, 平台监控的信息集中显示, 一目了然, 可通过图表展示某一时间的监控指标趋势, 并记录运维日志。

按照监控的范围, 常用的监控平台有如下几个, 当然还有很多没能罗列出来。

11.1.1 日志监控平台

(1) ELK

ELK 是一个开源的实时日志分析平台, 由 Elasticsearch、Logstash 和 Kibana 三个开源工具组成, 集中化管理日志, 监控系统日志、应用程序日志和安全日志等。运维人员可以根据日志了解服务器的配置负荷、系统性能和安全, 从而在错误发生时可以及时采取措施。ELK 适合做搜索日志, 但不适合做大数据统计, 在统计分析上效果不是很好。

(2) Splunk

Splunk 是一款商业版日志分析软件, 具有添加日志、生成图形化报表和非常强大的搜索功能。它能处理常规的日志格式, 设立日志的索引, 支持复杂的查询语句, 支持交叉查询。

11.1.2 硬件和应用监控平台

(1) Zabbix+Grafana

Zabbix 是一个开源的基于 Web 的性能监控解决方案, 提供分布式系统监视及网络监视, 可以监控服务器、Web 应用程序、数据库、网络设备等的性能和提供报警机制。Zabbix 可以结合 Grafana 显示更加绚丽的监控界面。

(2) Nagios+Cacti

Nagios 是一个开源的服务器和网络监控解决方案, 为服务器、交换机、应用程序和服务提供完整的监控和报警机制。可以通过插件 API 扩展开箱即用的功能。Nagios 侧重于系统状态正常与否, 能够通过邮件和短信发送警报, 但图形化有所欠缺。由于 Cacti 比较着重于直观数据的监控, 易于生成图形, 用来监控网络流量、CPU 使用率、硬盘使用率等很合适。两者结合使用, 取长补短是比较好的一个解决方案。



质量全面管控：从项目管理到容灾测试

11.2 Zabbix 简介

Zabbix 是一个成熟的、功能强大的监控平台，能够收集并监控网络上的设备，并将数据存储在数据库中以供将来使用。它有着很多优秀的特性，如下所示：

- (1) Zabbix 服务器可以在大多数 UNIX 类操作系统上运行。
- (2) Zabbix 是开源的，支持自定义开发。
- (3) 它可以自由地创建自定义脚本、列项和图表等。
- (4) 它提供集中式的网页界面，可以管理参数而不用修改文件。
- (5) 用户权限管理功能可以对用户进行分组，指定特定服务访问的可读或全部权限。
- (6) 对于 trigger 和 actions 设定监控参数的阈值。
- (7) 能够查看各监控指标的趋势图。
- (8) 在系统超负载或停止运作时，在远程服务器上执行 Shell 命令解决问题。

Zabbix 每个新的版本都有新的特性，详情请参考官网 <http://www.zabbix.com>。

Zabbix 的安装不是很复杂，配置才是重点，本节着重介绍 Zabbix 的架构以及如何配置和使用 Zabbix。

11.2.1 系统架构

Zabbix 由如下几个组件构成。

(1) Zabbix Server (服务器)：负责收集 Zabbix Agent 发送的配置和数据信息，整合并统计数据。

(2) Database Storage (数据存储)：用于存储所有配置信息和 Zabbix 收集的数据。

(3) Web UI (界面)：基于 Web 的界面，通常与 Zabbix Server 组件运行在同一台主机上。

(4) Zabbix Proxy (代理器)：可选组件，主要适用于分布监控环境中，代理 Zabbix Server 收集 Zabbix Agent 的监控数据并统一发往 Zabbix Server 端，可以分担 Zabbix Server 的压力。Zabbix Proxy 本身没有前端，也不存放数据，只是将 Zabbix Agent 发来的数据暂时存放，再提交给 Zabbix Server。



(5) Zabbix Agent (代理程序): 部署在被监控主机上, 负责主动收集客户端数据, 例如 CPU 负载、内存、硬盘使用情况, 并发往 Server 端或 Proxy 端。

Zabbix 的架构关系如图 11-1 所示。这里的 Zabbix Proxy 是可选的组件, Zabbix Agent 数量根据监控需要可以增减, 可以监控 CPU、内存、网络信息, 也可以通过 JMX 监控 Java 的应用程序。

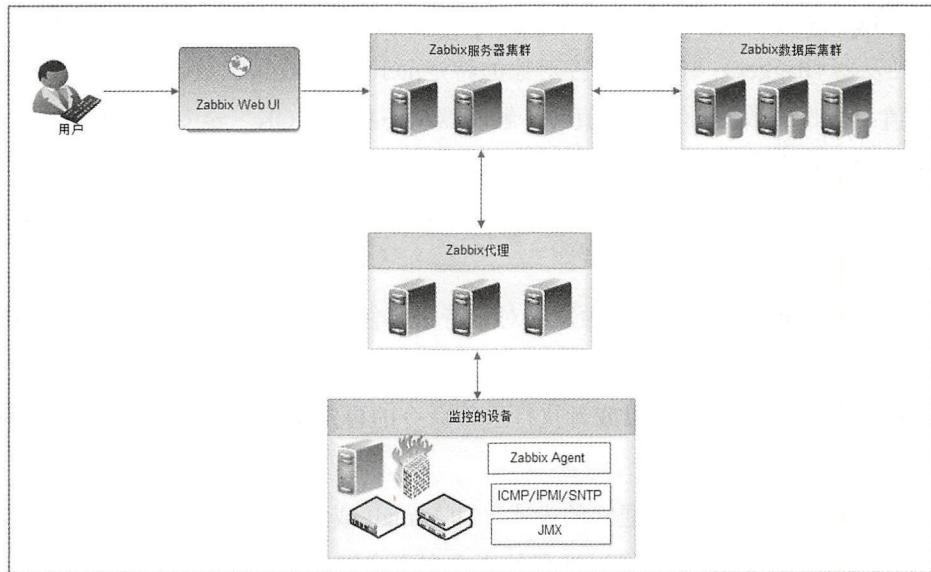


图 11-1 Zabbix 的架构

除了以上几个组件, Zabbix 常用的工具有如下几个。

(1) Get: 通常在 Server 端或者 Proxy 端执行, 是一个获取远程客户端信息的命令。例如, Server 端获取不到被监控端的 CPU 信息, 可以使用 `zabbix_get` 来排查。

(2) Sender: 用于发送数据给 Server 或者 Proxy, 通常用于耗时比较长的检查。在脚本执行完毕后, 使用 Sender 主动提交数据。

(3) Java 网关: Zabbix 2.0 之后引入的一个可选功能, 它能够主动获取数据, 最终提交数据至 Server 或者 Proxy。

以下是 Zabbix 中几个常见的术语说明, 仅供读者参考。

(1) 主机 (host): 被监控的服务器或网络设备, 可由 IP 或 DNS 名称指定。

(2) 主机组 (host group): 主机的逻辑容器, 可以包含主机和模板, 但同一个主机组



质量全面管控：从项目管理到容灾测试

内的主机和模板不能互相链接；主机组通常在给用户或用户组指派监控权限时使用。

(3) 监控项 (item)：一个特定监控指标的相关数据，这些数据来自于被监控对象。item 是 Zabbix 进行数据收集的核心，相对某个监控对象来说，每个 item 都由“key”标识。

(4) 应用 (application)：一组 item 的集合。

(5) 模板 (template)：定义被监控主机的条目集合，通常包含 item、trigger、graph、screen、application 和 low-level discovery rule，模板可以直接链接至某个监控主机。

(6) 触发器 (trigger)：评估某监控对象的特定 item 内接收到的数据是否在合理范围内，也就是阈值；接收的数据量大于阈值时，触发器状态将从“OK”转变为“Problem”，当数据再次恢复到合理范围时，又转变为“OK”。

(7) 事件 (event)：触发一个值得关注的事情，如触发器状态转变、新的 Agent 或重新上线的 Agent 的自动注册等。

(8) 动作 (action)：指 trigger 触发的对于特定事件事先定义的处理方法，如发送通知、何时执行操作等。

(9) 报警 (alert)：发送警报或者执行远程命令的自定义方案，如每隔 10 分钟发送一次警报，共发送 3 次等。

(10) 媒介 (media)：发送通知的方式，如 E-mail 或者 SMS 等。

(11) 通知 (notification)：通过选择的媒介向用户发送的事件信息。

(12) 前端 (frontend)：Zabbix 的 Web 界面。

(13) Web 场景 (web scenario)：用于检测 Web 站点可用性的 HTTP 请求。

(14) 远程命令 (remote command)：预定义的命令，对被监控主机远程调用命令。

11.2.2 配置 Zabbix

(1) 安装完 Zabbix 3.0 系统后，开始配置 Zabbix，首次在浏览器中打开 URL：http://your_server_ip/zabbix/setup.php，如图 11-2 所示，然后根据提示进行操作。

(2) 单击“Next step”按钮，会检查各预装项的版本和状态，如图 11-3 所示，状态显示为“OK”才可进入下一步。



图 11-2 Zabbix 初始设置页面

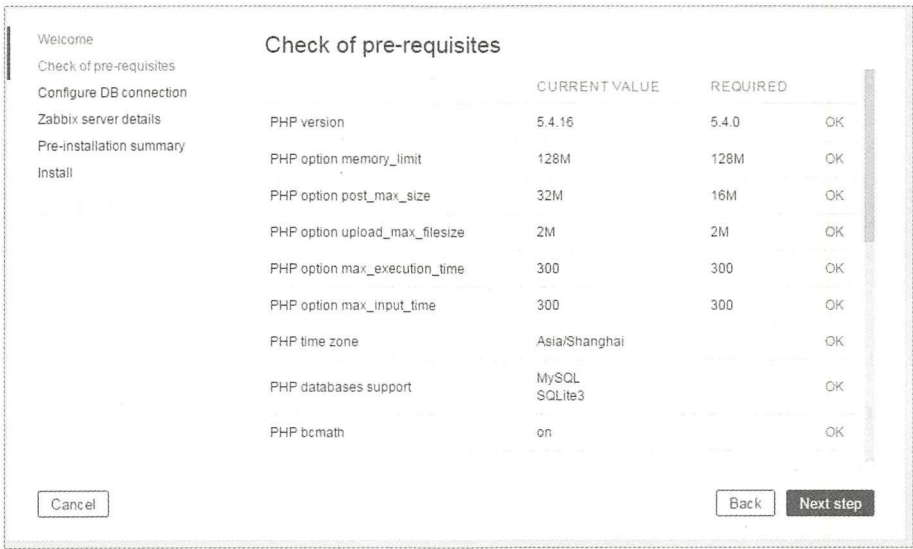


图 11-3 检查预装项

(3) 单击 “Next step” 按钮，配置 MySQL 数据库信息，如图 11-4 所示，输入数据库



的连接信息、用户名和密码。

- Database: 数据库类型, 如 MySQL。
- Database host: 数据库的主机地址, 可以是 IP 地址或主机名, 如 10.108.12.196 或者 localhost (使用本机的数据库)。
- Database port: 数据库端口, 输入 0 表示默认。
- Database name: 数据库的名称, 如 zabbix。
- User: 数据库的用户名, 如 zabbix。
- Password: 数据库的密码, 如 123456。

图 11-4 数据库连接配置

(4) 单击 “Next step” 按钮, 输入 Zabbix 服务器端信息, 如图 11-5 所示。

- Host: Zabbix Server 主机的 IP 地址或者主机名, 本机默认填写 localhost。
- Port: Zabbix 服务端口, 默认为 10051。
- Name: 安装名, 可选。

(5) 单击 “Next step” 按钮, 提示如图 11-6 所示的信息 “Congratulations!...”, 表明安装完成。



Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

Zabbix server details

Please enter the host name or host IP address and port number of the Zabbix server, as well as the name of the installation (optional).

Host

Port

Name

Cancel Back Next step

图 11-5 Zabbix 服务端信息

图 11-6 Zabbix 配置成功

(6) 最后单击“Finish”按钮，打开如图 11-7 所示的登录页面。输入默认用户名和密码 admin/zabbix 就可以登录了。

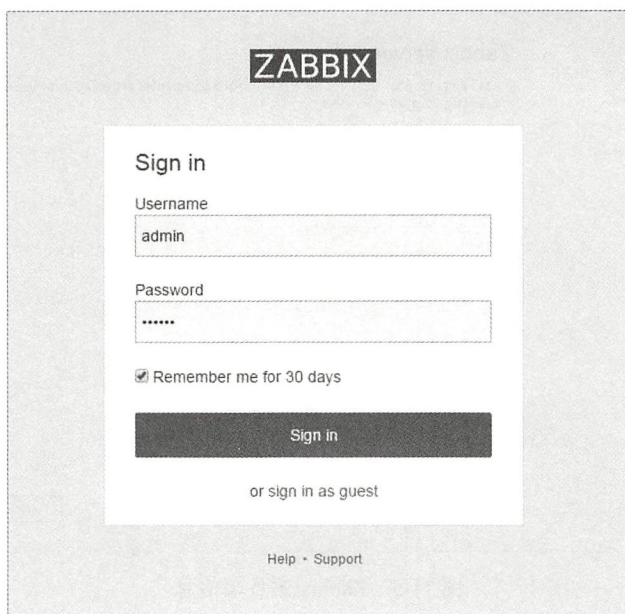


图 11-7 登录 Zabbix

(7) 由于 Zabbix 2.4 及以上版本默认关闭了中文支持，因此需要修改配置文件 `locales.inc.php`。在 Linux 系统下可以使用 `vim` 命令，其路径以实际安装路径为准，命令如下：

```
vim /var/www/html/zabbix/include/locales.inc.php
```

在打开的文件中，找到下面这行内容，把 “false” 改为 “true”：

```
'zh_CN' => array('name' => _('Chinese (zh_CN)'), 'display' => false),
```

然后用浏览器访问 Zabbix，如图 11-8 所示，选择菜单 “Administrator” → “Users”，选择一个用户 “Admin”，修改 “Language” 下拉列表框可设置中文显示，值为 “Chinese(zh_CN)”。

11.2.3 常见的配置问题

(1) PHP `mbstring.func_overload`：检测没通过。

在配置步骤 (2) 中，PHP `mbstring.func_overload` 显示 “off”，检测没通过，如图 11-9 所示。

图 11-8 修改为中文显示

2. Check of pre-requisites	PHP mbstring	on		OK	Close
	PHP mbstring.func_overload	on	off	Fail	
3. Configure DB connection	PHP sockets	on		OK	Close
	PHP gd	2.1.0	2.0	OK	

图 11-9 PHP mbstring.func_overload 检测没通过

解决方法是在 php.ini 文件中，修改 mbstring.func_overload 值为 0。

```
mbstring.func_overload=0
```

(2) Zabbix server is not running: 服务器未运行。

正常安装 Zabbix 并初始化配置完成后，登录 Zabbix，“Status of Zabbix”区域中“Zabbix server is running”行显示值为“No”，如图 11-10 所示。

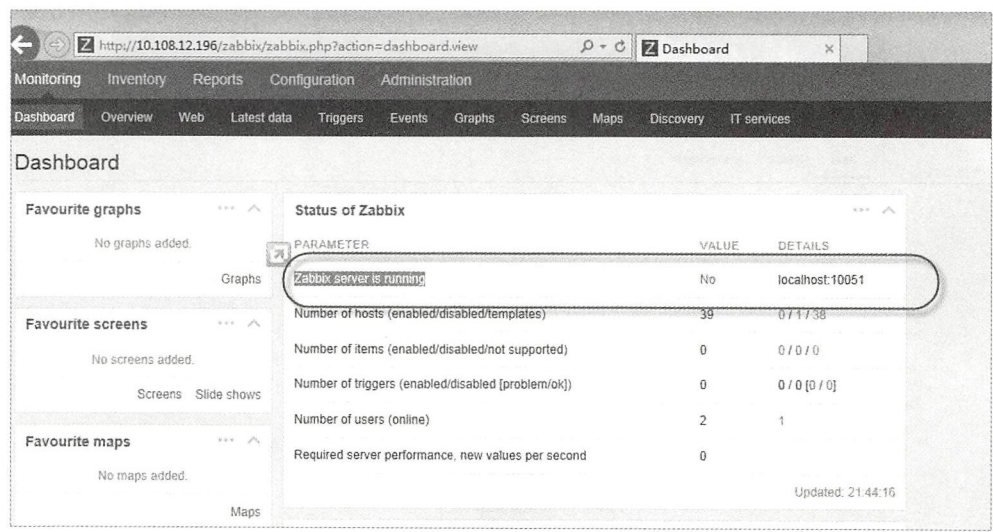


图 11-10 Status of Zabbix 报错

下面几种情况都有可能引起上面这个错误。

(1) SELinux 是否关闭。一定要关闭它，否则甚至连 Zabbix 的 discovery 功能也不能正常使用。

(2) 查看 Zabbix Server 的 Web 目录下的参数\$ZBX_SERVER 是否为 IP。

(3) 查看 PHP 的 Fsockopen 模块是否启用。

在 php.ini 文件中修改 allow_url_fopen 和 extension 值。

```
allow_url_fopen = On
extension=php_openssl.dll
```

(4) 查看 Web Server 启动状态，可以重启 Web Server 消除启动异常。

11.2.4 监控主机

(1) 选择“Configuration”→“Hosts”菜单项，在打开的页面中输入 Hostname、Groups、IP Address 和 Port，如图 11-11 所示。

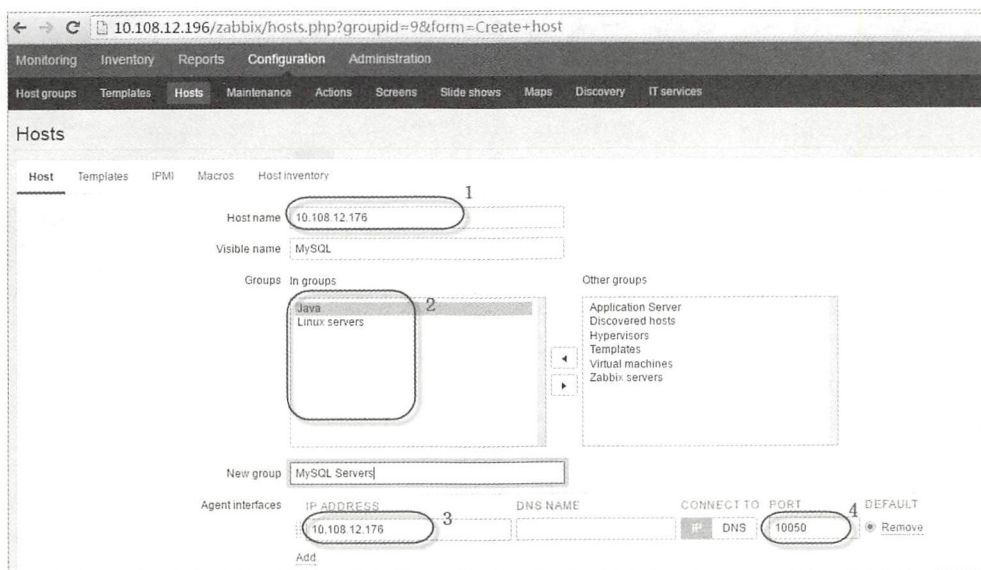


图 11-11 Hosts 页面设置

单击“Templates”选项卡，可以看到已经链接的模板显示在“Linked templates”中，如图 11-12 所示。也可以在页面中设置“Link new templates”关联更多模板。

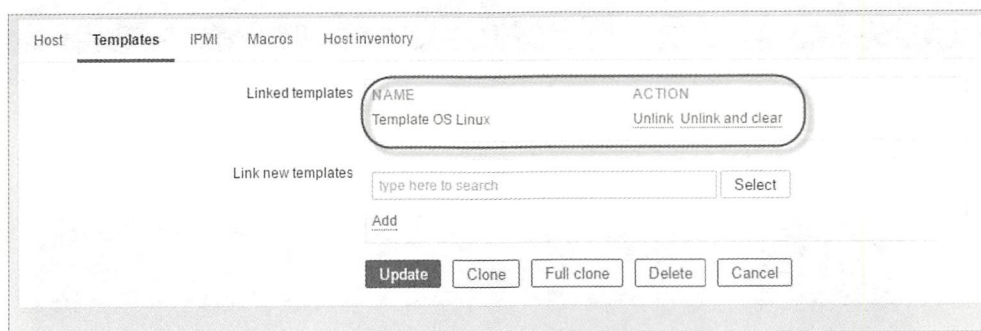


图 11-12 设置 Linked templates

(2) 添加模板后单击“Update”按钮确认添加，添加成功后，会显示如图 11-13 所示的列表。

在显示的列表中，“AVAILABILITY”列表项会显示各主机的连接状态，包括 ZBX、SNMP、JMX 和 IPMI，如图 11-14 所示，其中绿色表示已连接，红色表示未连接，灰色表示未配置。

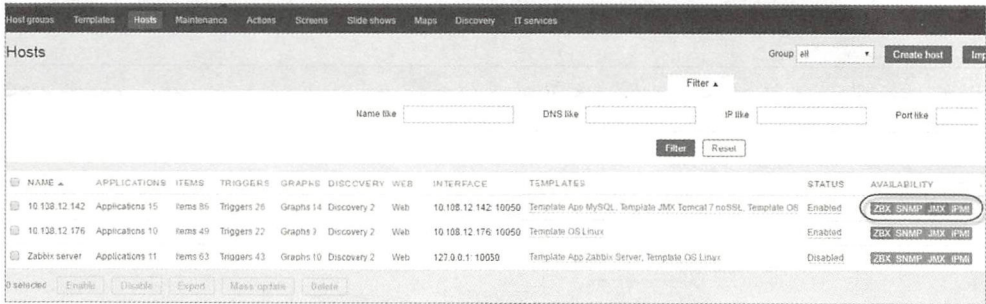


图 11-13 主机列表

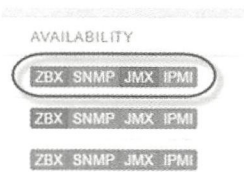


图 11-14 主机组件状态

(3) 主机添加成功后，选择“Monitoring”→“Graphs”菜单项，在打开的页面中可以选择 Group、Host 和 Graph 查看各监控项，还可以修改收集时间段，如图 11-15 所示。

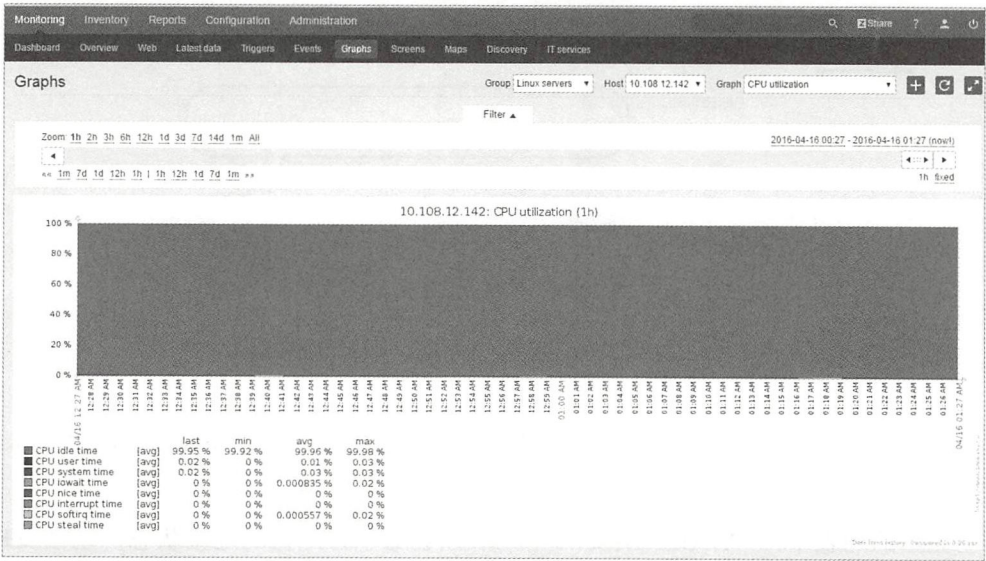


图 11-15 Graphs 选项页面

11.3 美化界面 Grafana

Grafana 是一款显示数据的开源的度量仪表盘和图形编辑器，它功能齐全，适合处理时序数据，能够支持 Graphite、Elasticsearch、Prometheus、InfluxDB、OpenTSDB 和 KairosDB。它丰富的图表展示非常交互式，可以自定义线条、点和条形等图表元素；它还提供两个主题风格，满足不同用户的需求；能够设置模板和变量，直接从数据库获得数据。本节来介绍下如何使用 Grafana-Zabbix 插件来获取 Zabbix 数据库的数据，如果想要提前感受一下 Grafana，可以访问 <http://play.grafana.org/>。如下是实例中系统和软件的版本要求。

- 操作系统：CentOS7。
- Zabbix：3.0.0 版本。
- Grafana：2.5.0.1 版本。
- Grafana-Zabbix 插件：2.5.0-1 版本。

11.3.1 部署 Grafana

(1) 下载 Grafana 安装包和 Grafana-Zabbix 插件，其下载地址分别为：

https://grafanarel.s3.amazonaws.com/builds/grafana-2.5.0-1.x86_64.rpm

<https://github.com/alexanderzobnin/grafana-zabbix/archive/v2.5.1.tar.gz>

提示：Grafana 的版本和 Grafana-Zabbix 版本必须匹配，否则会出现异常。

(2) 在 CentOS 系统中，使用 yum 安装 Grafana 的 rpm 包，命令如下：

```
sudo yum install grafana-2.5.0-1.x86_64.rpm
```

(3) 将 Grafana-Zabbix 插件的“zabbix”文件夹复制到 Grafana 目录的“plugins/datasource”文件夹下，执行如下命令：

```
cp -r grafana-zabbix/zabbix/ /usr/share/grafana/public/app/plugins/datasource/
```

(4) 启动 Grafana，运行 grafana-server 服务，执行如下命令：

```
sudo service grafana-server start
```

启动 Grafana 后，在浏览器中输入“<http://127.0.0.1:3000/login>”就可以看到页面，如图 11-16 所示。默认用户名为 admin，默认密码为 admin。

质量全面管控：从项目管理到容灾测试

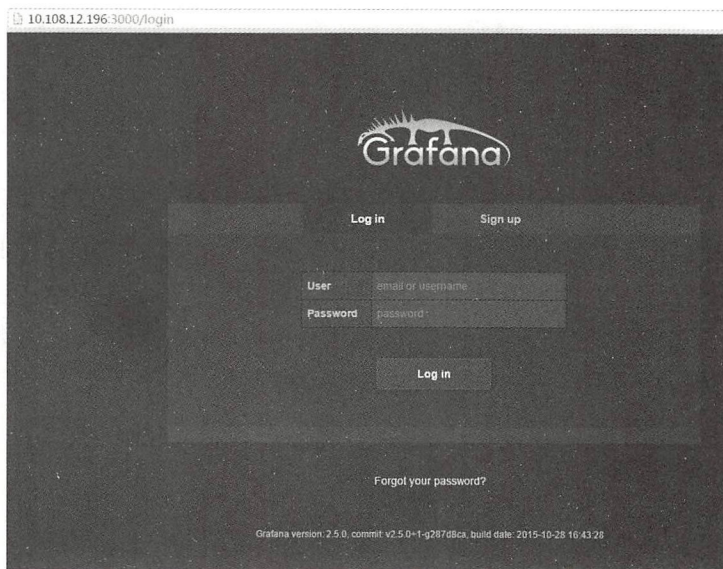


图 11-16 Grafana 登录页面

11.3.2 使用 Grafana

(1) 设置主题风格

登录 Grafana 之后，选择左侧菜单项的个人信息（Profile），打开个人信息维护界面，在“UI Theme”下拉列表框中可以设置 Grafana 主题：dark 或者 light。如图 11-17 所示是 dark 主题风格。

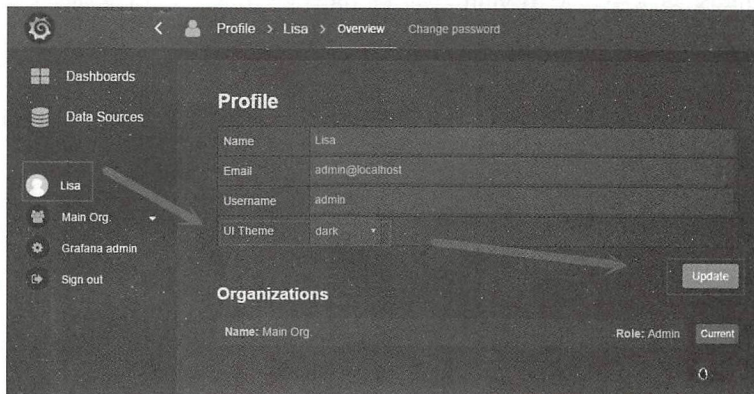


图 11-17 dark 主题风格

如图 11-18 是 light 主题风格。

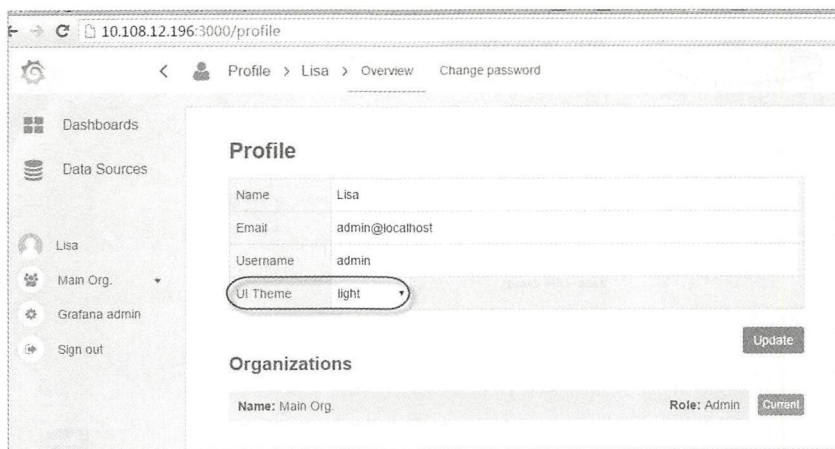


图 11-18 light 主题风格

(2) 配置数据源

登录 Grafana 后，单击左侧的“Data Sources”选项，然后单击页面上方的“Add new”选项，如图 11-19 所示。其中，各选项说明如下。

- **Name:** 数据源名称，可选取任意有意义的名称。
- **Type:** 数据源类型，支持 CloudWatch、Elasticsearch、Graphite、InfluxDB 0.9.x、InfluxDB 0.8.x、KairosDB、OpenTSDB、Prometheus 和 Zabbix。为了显示 Zabbix 的监控数据，这里输入值为“Zabbix”。
- **Url:** Zabbix 的 API 地址，如“http://127.0.0.1/zabbix/api_jsonrpc.php”，其中 IP 地址是 zabbix-server 服务器的 IP 地址。
- **Access:** 如果使用 Zabbix Proxy，则选择“proxy”，否则选择“direct”。
- **User/Password:** Zabbix 的 API 账号和密码是 Zabbix 的 Web 端的登录账号和密码，默认账号为 admin，默认密码为 zabbix。

设置完成后，单击“Add”按钮，新建一个数据源。

单击“Test Connection”按钮可以测试是否连接成功，如图 11-20 所示。如果提示“Success Zabbix API version: 3.0.0”，说明 Zabbix 数据源连接成功。



质量全面管控：从项目管理到容灾测试

The screenshot shows the Grafana interface with the 'Data Sources' menu item highlighted in the left sidebar (labeled '1'). The main content area is titled 'Add data source' and contains the following fields:

- Name:** zabbix (with a 'Default' checkbox)
- Type:** Zabbix (dropdown menu)
- Http settings:**
 - Url:** http://127.0.0.1/zabbix/api_jsonrpc.php
 - Access:** proxy (dropdown menu)
 - Basic Auth:** Enable (checkbox)
- Zabbix API details:**
 - User:** admin
 - Password:** (masked with dots)
 - Trends:** Enable (checkbox)
 - Metrics limit:** 100

An 'Add' button is located at the bottom right of the form.

图 11-19 配置 Grafana 数据源

The screenshot shows the 'Edit data source' page for the 'zabbix' data source. The fields are identical to the previous screenshot, but the 'Access' dropdown is now set to 'direct'. Below the configuration fields is a 'Test results' section showing a 'Success' message: 'Zabbix API version: 3.0.0'. At the bottom, there are three buttons: 'Save', 'Test Connection' (highlighted with a red box and an arrow), and 'Cancel'.

图 11-20 测试“Zabbix 数据源”





(3) 新建面板

在页面左侧单击“Dashboards”选项，然后在页面上方单击“Home”选项，再单击“New”按钮，即可新建一个面板，如图 11-21 所示。

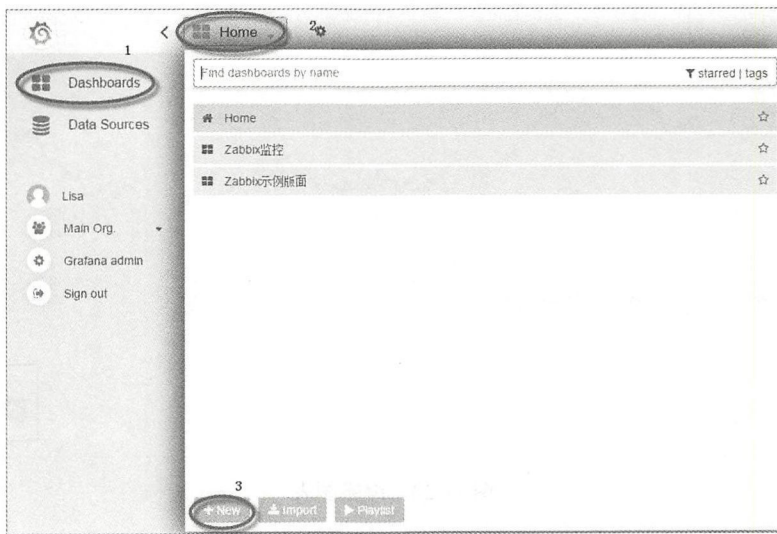



图 11-21 新建面板

(4) 添加 Graph

在打开的面板中，单击页面中间的图标，再选择“Add Panel”→“Graph”选项，如图 11-22 所示，即可添加图表。

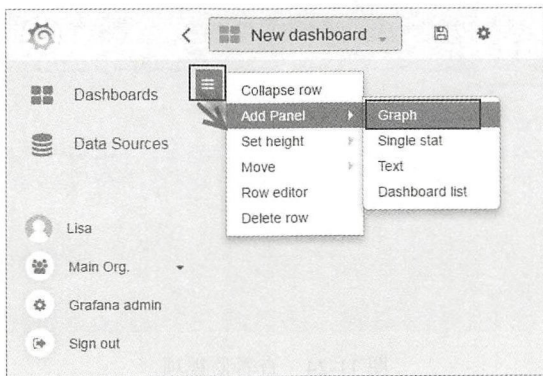


图 11-22 新建图表





质量全面管控：从项目管理到容灾测试

选中图表，单击“Edit”选项，会显示图表设置，右下角的图标用于选择之前创建的 Zabbix 数据源，如图 11-23 所示。

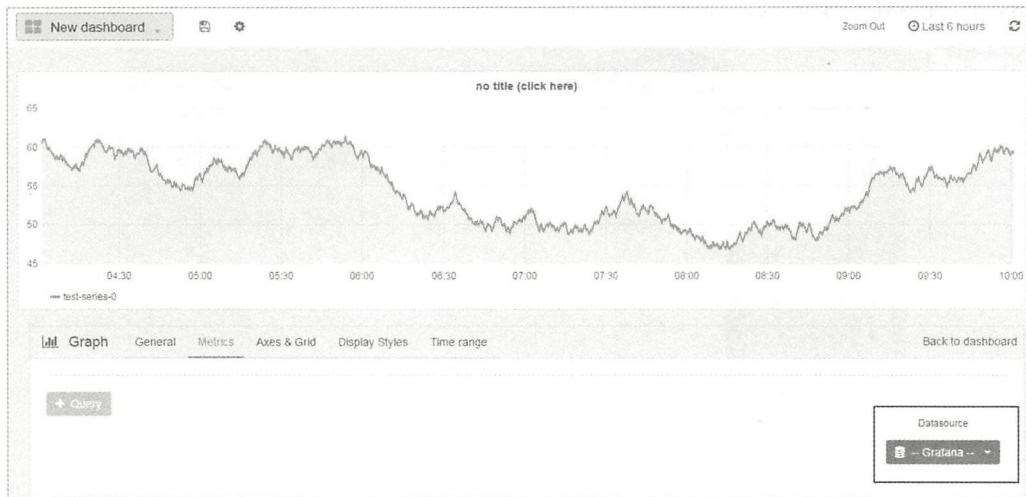


图 11-23 编辑图表

单击图表下方的“Metrics”选项卡，选择所需要监控的 Group、Host、Application 和 Item，即可看到对应的监控项，如图 11-24 所示。

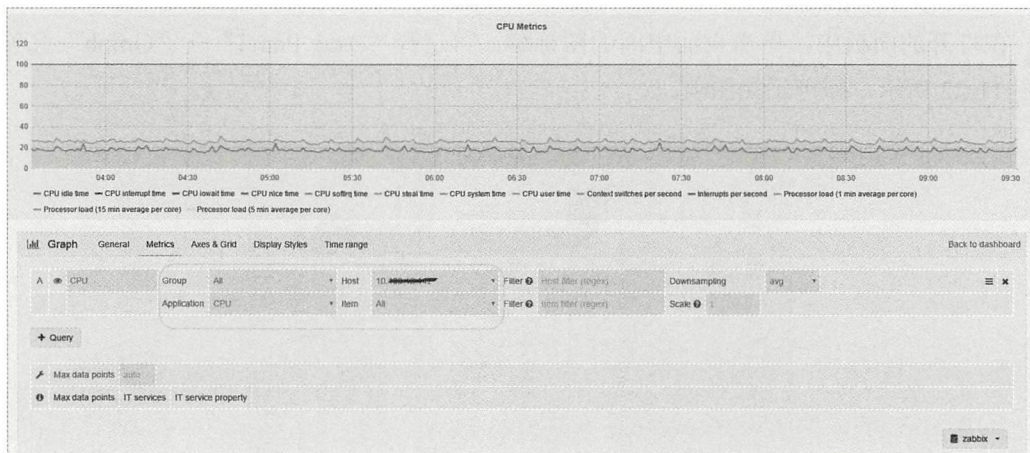
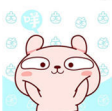


图 11-24 查看监视项





11.4 服务器故障排查

在项目开发和运维过程中，服务器或多或少会出现故障，如服务器断电、网络连接丢失、应用服务器无响应、磁盘容量已满或 CPU 使用率过高等问题。服务器问题的原因很少可以一下就想到，那么如何排查处理呢？基本上都需要从如下步骤入手，结合多个特征推断和验证问题的根源。

11.4.1 清楚故障的前因后果

服务器出现故障，首先保持心态不要心慌，弄明白这台服务器故障的具体情况。

- What? 故障的表现是无响应还是报错，最后一次更新什么内容，如代码、数据库等。
- When? 故障发生的时间。
- Who? 故障影响的特定用户群，如已登录用户、已退出用户或者某个地域的用户等。

除了以上的 3 个 W 问题，还应该关注如下几点。

- 故障是否可重现。
- 有没有故障出现的规律，比如每一天服务器宕机一次。
- 是否有监控平台可用，如 InfluxDB、Zabbix 和 Nagios 等。
- 是否有日志可以查看，如 ELK、Splunk、Loggly、Airbrake 和 Graylog 等。

11.4.2 搜寻蛛丝马迹

服务器故障排查的第一步就是尽可能地搜集信息，查看当前有哪些在线用户、历史访问用户和历史执行命令，可以提早发现一些异常的活动。下面是几个查看命令。

(1) 查看当前的在线用户，命令如下：

```
w 或者 who
```

(2) 查看历史访问的用户及 IP 地址，命令如下：

```
last
```

(3) 显示系统已经运行了多长时间，命令如下：





质量全面管控：从项目管理到容灾测试

uptime

如图 11-25 所示的实例中，可以发现当前服务器有 2 个 root 用户会话在线，其中一个会话从客户端 10.108.5.212 远程访问，另一个会话直接通过 tty 访问。

```
[root@localhost ~]# w
15:49:32 up 36 days, 2:05, 2 users, load average: 0.00, 0.01, 0.05
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
root      tty1     -               25Jan16 13days 0.09s   0.09s -bash
root      pts/0    10.108.5.212    15:10    4.00s   0.01s   0.00s w

[root@localhost ~]# last
root      pts/0    10.108.5.212    Tue Mar 1 15:10    still logged in
root      pts/0    10.108.5.212    Tue Feb 16 19:07 - 18:42 (2+23:34)
root      pts/2    10.108.5.212    Mon Jan 25 22:19 - 01:17 (7+02:57)
root      pts/1    10.108.5.212    Mon Jan 25 19:10 - 19:53 (9+00:42)
root      pts/1    10.108.5.212    Mon Jan 25 19:09 - 19:10 (00:00)
root      pts/0    10.108.5.212    Mon Jan 25 18:28 - 01:17 (7+06:48)
root      tty1     -               Mon Jan 25 18:20    still logged in
reboot    system boot  3.10.0-229.el7.x Mon Jan 25 18:11 - 15:49 (35+21:38)
root      tty1     -               Mon Dec 7 10:03    - crash (49+08:07)
```

图 11-25 w 和 last 的输出信息

(4) 遍寻历史执行命令

查看服务器执行过的历史命令，如果需要显示命令的执行时间，就得更新 HISTTIMEFORMAT 环境变量。设置完 HISTTIMEFORMAT 环境变量后，只有新执行的 bash 命令才会被打上正确的时间戳。环境变量可以在文件/etc/profile 的最后添加如下部分来设置：

```
USER_IP=`who -u am i 2>/dev/null| awk '{print $NF}'|sed -e 's/[()]/g'`
export HISTTIMEFORMAT="%F %T"["whoami"][${USER_IP}] "
```

接着执行如下的命令，使环境变量生效：

source /etc/profile

使用如下命令可以显示服务器上执行过的历史命令，结果输出如图 11-26 所示，显示历史执行命令、来源和时间：

history|more

```
[root@localhost ~]# USER_IP=`who -u am i 2>/dev/null| awk '{print $NF}'|sed -e 's/[()]/g'`
[root@localhost ~]# export HISTTIMEFORMAT="%F %T"["whoami"][${USER_IP}] "
[root@localhost ~]# history|more
1 [2016-03-01 16:02:51] [root] [10.108.5.212] ifconfig
2 [2016-03-01 16:02:51] [root] [10.108.5.212] halt
3 [2016-03-01 16:02:51] [root] [10.108.5.212] service httpd restart
4 [2016-03-01 16:02:51] [root] [10.108.5.212] /etc/init.d/zabbix_agentctl start
5 [2016-03-01 16:02:51] [root] [10.108.5.212] /etc/init.d/zabbix_agent start
6 [2016-03-01 16:02:51] [root] [10.108.5.212] /etc/init.d/zabbix_agentd start
7 [2016-03-01 16:02:51] [root] [10.108.5.212] netstat -nplut |grep zabbix
8 [2016-03-01 16:02:51] [root] [10.108.5.212] hostname
9 [2016-03-01 16:02:51] [root] [10.108.5.212] where is yum
10 [2016-03-01 16:02:51] [root] [10.108.5.212] cd /etc/yum.repos.d/
11 [2016-03-01 16:02:51] [root] [10.108.5.212] ls
```

图 11-26 查询历史执行命令





11.4.3 列出当前运行的进程

(1) 使用 `ps` 命令查看进程树，以树状显示正在运行的进程，树的根节点为 `pid` 或 `init`。如果指定了用户名，进程树将以用户所拥有的进程作为根节点。命令格式如下：

```
ps tree -ap
```

- `-a` 参数：相同名称的进程不合并显示，并且会显示命令行参数。
- `-p` 参数：同时显示每个进程的 PID。

由于 `ps` 输出的信息比较多，最好与 `more` 或 `less` 命令配合使用，输出结果如图 11-27 所示。

```
[root@localhost ~]# ps tree -ap | less
systemd,1 --system --deserialize 21
├─NetworkManager,682 --no-daemon
│   ├─dhclient,1470 -d -q -sf /usr/libexec/rm-dhcp-helper -pf /var/run/dhclient-eno16780032.pid -lf...
│   └─{NetworkManager},729
│       └─{NetworkManager},732
│           └─{NetworkManager},734
├─auditd,570 -n
│   └─{auditd},581
├─crond,2872 -n
├─dbus-daemon,600 --system --address=systemd: --nofork --nopidfile --systemd-activation
│   └─{dbus-daemon},613
├─grafana-server,12773 --config=/etc/grafana/grafana.ini --pidfile= cfg:default.paths.logs=/var/log/grafana:default.paths.data:
│   └─{grafana-server},12775
│       └─{grafana-server},12776
│           └─{grafana-server},12777
│               └─{grafana-server},12778
│                   └─{grafana-server},14162
└─httpd,18264 -DFOREGROUND
```

图 11-27 查看进程树

(2) 使用 `ps` 命令查看进程列表，可以按 CPU 使用率排序显示，或者过滤显示某个进程，命令如下。

```
ps aux --sort=%cpu
```

也可以用类似命令查看内存，使用 `--sort=%mem` 选项，此命令可以实时监控系统资源信息。

命令输出结果如图 11-28 所示，显示了进程号为 498 的进程的 CPU 使用率最高，值为 0.3%。

root	27261	0.0	0.1	115488	2132	pts/1	Ss	Mar01	0:00	-bash
mysql	27414	0.0	0.0	115244	1684	?	Ss	Jan25	0:00	/bin/sh /usr/bin/mysqld_saf
mysql	27565	0.1	13.3	1009184	250960	?	Ss	Jan25	78:30	/usr/sbin/mysqld --basedir=
n-d										
root	498	0.3	0.1	121588	3464	pts/1	S+	Mar02	5:07	htop -p 15323

图 11-28 查看最高 CPU 使用率

也可以使用如下命令查看占用 CPU 最高的进程，输出结果如图 11-29 所示：





质量全面管控：从项目管理到容灾测试

```
ps aux|head -1;ps aux|grep -v PID|sort -rn -k +3|head
```

```
[root@localhost ~]# ps aux|head -1;ps aux|grep -v PID|sort -rn -k +3|head
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      498  0.3  0.1 121588 3464 pts/1    S+   Mar02   5:08 htop -p 15323
mysql     27565  0.1 13.3 1009184 250960 ?        S1   Jan25   78:30 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plu
n-dir=/usr/lib64/mysql/plugin --log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mys
sock
zabbix    15470  0.0  0.0 2858056 1716 ?        S    Feb18   0:39 /usr/local/sbin/zabbix_server: self-monitoring [processed data
n 0.000008 sec, idle 1 sec]
zabbix    15469  0.0  0.1 2858056 1944 ?        S    Feb18   0:13 /usr/local/sbin/zabbix_server: proxy poller #1 [exchanged data
ith 0 proxies in 0.000009 sec, idle 5 sec]
```

图 11-29 查看占用 CPU 使用率最高的进程

使用如下命令查看占用内存最高的进程，输出结果如图 11-30 所示：

```
ps aux|head -1;ps aux|grep -v PID|sort -rn -k +4|head
```

```
[root@localhost ~]# ps aux|head -1;ps aux|grep -v PID|sort -rn -k +4|head
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mysql     27565  0.1 13.3 1009184 250960 ?        S1   Jan25   78:56 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysq
/run/mysqld/mysqld.pid --socket=/var/lib/mysql/mysql.sock
zabbix    15384  0.0  6.8 2859348 129196 ?        S    Feb18   0:35 /usr/local/sbin/zabbix_server: poller #29 [got 0 values in 0.000002 sec, idle 1 sec]
zabbix    15382  0.0  6.8 2859352 128424 ?        S    Feb18   0:36 /usr/local/sbin/zabbix_server: poller #27 [got 0 values in 0.000003 sec, idle 1 sec]
zabbix    15380  0.0  6.8 2859352 128424 ?        S    Feb18   0:35 /usr/local/sbin/zabbix_server: poller #25 [got 0 values in 0.000002 sec, idle 1 sec]
zabbix    15379  0.0  6.8 2859348 128772 ?        S    Feb18   0:33 /usr/local/sbin/zabbix_server: poller #24 [got 0 values in 0.000002 sec, idle 1 sec]
```

图 11-30 查看占用内存最高的进程

如果发现某个进程的 CMD 后面接上<defunct>时，就代表该进程是僵尸进程，例如以下是一条进程信息：

```
1 2598 2598 2598 ? -l Ss 0 0:00 /usr/sbin/hcid<defunct>
```

当系统不稳定的时候就容易造成所谓的僵尸进程，可能是因为程序逻辑或算法的缺陷，或者用户操作不当造成的。如果系统出现很多僵尸进程，一定要找出父进程，跟踪优化主机环境。

命令“ps -Lf <PID>”还可以查看进程启动哪些线程，实例如图 11-31 所示。

```
[root@localhost ~]# ps -Lf 15323
UID      PID  PPID  LWP  C  NLWP  STIME  TTY      STAT  TIME
root     15323  1 15323  0  19  Feb18 ?        S1    0:00
root     15323  1 15324  0  19  Feb18 ?        S1    0:02
root     15323  1 15325  0  19  Feb18 ?        S1    0:45
root     15323  1 15326  0  19  Feb18 ?        S1    0:00
root     15323  1 15327  0  19  Feb18 ?        S1    0:00
root     15323  1 15328  0  19  Feb18 ?        S1    0:00
root     15323  1 15329  0  19  Feb18 ?        S1    0:01
root     15323  1 15330  0  19  Feb18 ?        S1    0:01
root     15323  1 15331  0  19  Feb18 ?        S1    0:00
root     15323  1 15332  0  19  Feb18 ?        S1    8:40
```

图 11-31 进程启动的线程

pstack 命令“pstack <PID>”会显示每个进程的栈跟踪，包括进程下面的各线程的栈信息，其线程号与“ps -Lf <PID>”中查询的结果是一致的，输出结果如图 11-32 所示。




```

[root@localhost ~]# pstack 15323
Thread 19 (Thread 0x7fd3a06c2700 (LWP 15324)):
#0 0x00007fd39fba2bcd in poll () from /lib64/libc.so.6
#1 0x00007fd39c9ff4d33 in NET_Timedout () from /usr/local/jdk1.7.0_79/jre/lib/amd64/libnet.so
#2 0x00007fd39c9ff60a7 in java_java_net_PlainSocketImpl_socketAccept () from /usr/local/jdk1.7.0_79/jre/lib/amd64/libnet.so
#3 0x00007fd395012d78 in ?? ()
#4 0x00000000e7a43578 in ?? ()
#5 0x00000000e7a0e3d0 in ?? ()
#6 0x0000000000000001 in ?? ()
#7 0x00007fd3a06c17e8 in ?? ()
#8 0x0000000000000000 in ?? ()
Thread 18 (Thread 0x7fd39da1b700 (LWP 15325)):
#0 0x00007fd3a029ea82 in pthread_cond_timedwait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
#1 0x00007fd39f45dc0f in os::PlatformEvent::park(long) () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#2 0x00007fd39f41fb2e in Monitor::Iwait(Thread*, long) () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#3 0x00007fd39f42000e in Monitor::wait(bool, long, bool) () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#4 0x00007fd39f5e78b9 in VMThread::loop() () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#5 0x00007fd39f5e7bc0 in VMThread::run() () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#6 0x00007fd39f45eca8 in java_start(Thread*) () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#7 0x00007fd3a029adc5 in start_thread () from /lib64/libpthread.so.0
#8 0x00007fd39fbad21d in clone () from /lib64/libc.so.6
Thread 17 (Thread 0x7fd39d91a700 (LWP 15326)):
#0 0x00007fd3a029ea82 in pthread_cond_wait@@GLIBC_2.3.2 () from /lib64/libpthread.so.0
#1 0x00007fd39f45d543 in os::PlatformEvent::park() () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#2 0x00007fd39f44cd4d in ObjectMonitor::wait(long, bool, Thread*) () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#3 0x00007fd39f2bc288 in JVM_MonitorWait () from /usr/local/jdk1.7.0_79/jre/lib/amd64/server/libjvm.so
#4 0x00007fd395012d78 in ?? ()
#5 0x0000000000000000 in ?? ()

```

图 11-32 pstack 栈跟踪

11.4.4 监听网络服务

如果进程和线程栈无明显的异常情况，接下来可以查看异常是否是由网络服务引起的。分开执行如下 3 条命令，分别显示出 TCP、UDP 和所有的监听网络连接。

```

netstat -ntlp
netstat -nulp
netstat -nxlp

```

如果要显示所有存在的连接，netstat 会比较慢，可以先用如下 ss 命令查看一下总体情况，显示处于活动状态的套接字信息。从中可以看出在不同连接状态下 TCP 连接时间的设置。

```
ss -s
```

如果怀疑网络有性能问题，可以使用 Netperf 工具。它根据应用的不同，可以进行不同模式的网络性能测试，即批量数据传输模式和请求/应答模式。Netperf 测试结果所反映的是一个系统能够以多快的速度向另外一个系统发送数据，以及另外一个系统能够以多快的速度接收数据。

11.4.5 查看硬件状态

有时，服务器响应慢是由硬件问题引起的，对于硬件状态的监控，需要查看服务器的

CPU、内存、总线设备和 I/O 等情况。通过综合分析，定位故障瓶颈。

- 检查磁盘使用量：服务器硬盘是否已满。
- 是否开启了 swap 交换模式（si/so）。
- CPU 占用情况：占用高 CPU 时间片的是系统进程还是用户进程。

1. CPU 和内存

按顺序执行如下两个命令可以查看内存和 CPU 信息：

```
free -m  
top 或者 htop
```

观察是否出现如下问题：

- 还有多少空余的内存？服务器是否正在内存和硬盘之间进行交换？
- CPU 使用率有多高？服务器是几核的处理器？是否某些 CPU 核的负载过多？
- 服务器最大的负载来自哪个进程？平均负载是多少？

下面简要地讲解以上两个命令的使用方法。

(1) 第一个 free 命令可以查看空闲的内存容量，格式如下：

```
free [-b|-k|-m|-g] [-t] [-s <间隔秒数>]
```

其参数说明如下。

- -b：以 Byte 为单位显示内存使用情况。
- -k：以 KB 为单位显示内存使用情况。
- -m：以 MB 为单位显示内存使用情况。
- -g：以 GB 为单位显示内存使用情况。
- -o：不显示缓冲区调节列。
- -s: <间隔秒数>：持续观察内存使用状况，如“-s 10”表示每隔 10 秒查看一次内存。
- -t：在输出的最终结果中显示物理内存与交换空间的容量。

本实例中显示了目前系统的内存容量，单位以 MB 显示，输出结果如图 11-33 所示。

```
[root@localhost ~]# free -m
```

	total	used	free	shared	buff/cache	available
Mem:	1840	535	88	140	1215	938
Swap:	2047	2	2045			

图 11-33 free 的内存容量输出

在实例的输出结果中，Mem 一行显示了物理内存的容量，Swap 一行则是虚拟内存的容量。total 列是总容量，used 列是已被使用的容量，free 列是空闲的容量，shared 是共享的容量，buff/cache 列表示缓冲及缓存的容量，available 列则是真正空闲可以使用的容量。

实例中显示的物理内存 Mem free 值为 88，看起来几乎耗尽。不过，服务器仍然有 938MB 容量是可用的，也就是说，系统最大程度地利用全部内存，目的是为了让系统的存取效率更好。这里特别需要注意的是 Swap 值。一般来说，Swap free 最好不要被使用，尤其是 Swap used 的值最好不要超过 20%。

(2) 第二个命令可以查看各个进程的 CPU 和内存等信息。

那么应该选择 top 还是 htop? top 命令默认每 3 秒刷新一次结果，也可以修改为默认每 5 秒刷新一次，如“top -d 5”。与 top 相比，htop 启动更快，支持鼠标操作，可以横向或纵向滚动浏览进程列表，结束进程时不需要输入进程号。在 htop 打开的界面中，可以使用鼠标单击列名，如 CPU% 进行排序，如图 11-34 所示。

CPU		Mem		Swap		Tasks: 138, 118 thr, 3 running		Load average: 0.00 0.03 0.05		Uptime: 36 days, 21:39:54	
PID	USER	PRI	NI	VERT	RES	SHR	S	CPU%	MEM%	TIME+	Command
27565	mysqld	20	0	985M	244M	580	S	0.0	13.3	1:16:04	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27582	mysqld	20	0	985M	244M	580	S	0.0	13.3	8:50:90	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27567	mysqld	20	0	985M	244M	580	S	0.0	13.3	3:42:15	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27574	mysqld	20	0	985M	244M	580	S	0.0	13.3	2:51:40	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27572	mysqld	20	0	985M	244M	580	S	0.0	13.3	2:42:08	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27573	mysqld	20	0	985M	244M	580	S	0.0	13.3	2:41:41	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27575	mysqld	20	0	985M	244M	580	S	0.0	13.3	2:39:07	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15448	mysqld	20	0	985M	244M	580	S	0.0	13.3	2:08:80	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15510	mysqld	20	0	985M	244M	580	S	0.0	13.3	1:54:35	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27580	mysqld	20	0	985M	244M	580	S	0.0	13.3	1:44:6	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15537	mysqld	20	0	985M	244M	580	S	0.0	13.3	1:43:13	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
27578	mysqld	20	0	985M	244M	580	S	0.0	13.3	1:40:16	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15503	mysqld	20	0	985M	244M	580	S	0.0	13.3	0:49:65	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15514	mysqld	20	0	985M	244M	580	S	0.0	13.3	0:48:57	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15507	mysqld	20	0	985M	244M	580	S	0.0	13.3	0:48:25	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr
15515	mysqld	20	0	985M	244M	580	S	0.0	13.3	0:47:82	usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr

图 11-34 htop 输出结果

提示：如果某个进程的 CPU 利用率一直维持在 100%，则说明这个线程可能有死循环；如果某个进程的 CPU 利用率一直处于 TOP10 的位置，则说明这个线程可能有性能问题，还需结合 pstack 查看线程栈信息。

(3) 其实还可以使用 mpstat 命令来统计每个 CPU 处理器的使用状态统计，如下命令每 2 秒收集一次 CPU 信息，总共 10 次，输出结果如图 11-35 所示。有些程序可能会导致一个 CPU 过载，而其他 CPU 却很空闲，可以通过 mpstat 诊断这类问题：

```
mpstat 2 10
```

```
[root@localhost ~]# mpstat 2 10
Linux 3.10.0-229.el7.x86_64 (localhost.localdomain) 08/02/2016 _x86_64(
1 CPU)
```

		%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	
11:59:10 PM	CPU									
	%gnice	%idle								
11:59:12 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	100.00								
11:59:14 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	100.00								
11:59:16 PM	all	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 99.50
11:59:18 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 100.00
11:59:20 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 100.00
11:59:22 PM	all	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00 99.50
11:59:24 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 100.00
11:59:26 PM	all	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 99.50
11:59:28 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 100.00
11:59:30 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00 100.00
Average:	all	0.10	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00 99.85

图 11-35 mpstat 输出结果

2. 总线设备

根据如下三个命令可以了解硬件问题的来源和性能改进的办法。

- (1) 使用命令 `lspci`，显示系统中所有 PCI 总线设备或连接到该总线上的所有设备。
- (2) 也可以使用命令 `dmidecode|more` 查看 DMI 信息，即 Desktop Management Interface。其中，`dmidecode` 命令的各参数说明如下。

- `-h`：查看帮助信息。
- `-q`：不显示未知设备。
- `-t type`：查看指定类型的信息，如 `bios`、`system`、`memory` 和 `processor` 等。
- `-s keyword`：查看指定的关键字的信息，如 `system-manufacturer`、`system-product-name`、`system-version` 和 `system-serial-number` 等。

如下实例可以查看设备内存槽数，每个槽位插入多少大小的内存。

```
[root@localhost ~]# dmidecode|grep -P -A5 "Memory Device" |grep Size
Size: 2048 MB
Size: No Module Installed
Size: No Module Installed
```

如下实例可以查看设备最大支持的内存容量。

```
[root@localhost ~]# dmidecode -t memory |grep "Maximum Capacity"
Maximum Capacity: 1 TB
```

- (3) 使用 `ethtool` 查询和设置网卡参数。查看网卡是否正确设置、是否正运行半双工状态、网络传输速度是多少、是否传输或读写错误。

统性能是不会受到影响的。

- buff: 作为 buffer cache（缓冲区缓存）的内存，对块设备的读写进行缓冲。
- cache: 作为 page cache（页面缓存）的内存，文件系统的缓存。如果缓存值大，则说明缓存的文件数多；如果频繁访问到的文件都能被缓存，那么磁盘的读 io bi（发送到块设备的块数）会非常小。
- si: 交换内存使用，由磁盘调入内存。
- so: 交换内存使用，由内存调入磁盘。

提示：内存够用时，si 和 so 的值都是 0；如果这两个值长期大于 0，那么系统性能就会受到影响，磁盘、I/O 和 CPU 资源都会被消耗完。

- bi: 从块设备读入的数据总量（读磁盘），单位为 KB/s。
- bo: 写入到块设备的数据总量（写磁盘），单位为 KB/s。

提示：随机磁盘读写的时候，这 2 个值越大（如超出 1M），能看到 CPU 在 I/O 等待的值也会越大。

- in: 每秒产生的中断次数。
- cs: 每秒产生的上下文切换次数。

提示：in 和 cs 的值越大，由内核消耗的 CPU 时间会越多。

- us: 用户进程消耗的 CPU 时间百分比。如果 us 的值比较高，则说明用户进程消耗的 CPU 时间多，但是如果长期超过 50%，那么就该考虑优化程序算法或者进行加速。
- sy: 内核进程所消耗的 CPU 时间百分比。如果 sy 的值比较高，说明系统内核消耗的 CPU 资源多，这并不是良性的表现，应该检查原因。
- id: CPU 处在空闲状态时间的百分比。
- wa: I/O 等待消耗的 CPU 时间百分比。如果 wa 的值比较高，则说明 I/O 等待比较严重，这可能是磁盘大量做随机访问造成的，也有可能是磁盘的带宽出现瓶颈，如块操作等。

(3) 查看当前占用 I/O 资源最高的进程信息，输出结果如图 11-38 所示。

```
dstat --top-io --top-bio
```

```
[root@localhost ~]# dstat --top-io --top-bio
-----most-expensive-----
i/o process          block i/o process
mysqld                20B 5399B  mysqld                68B 11k
sshd: root@          145B 240B
sshd: root@           47B 144B
sshd: root@           47B 144B
sshd: root@           47B 144B
sshd: root@           47B 144B
sshd: root@           47B 144B
sshd: root@           47B 144B
sshd: root@           47B 144B
sshd: root@           47B 144B
mysqld                 0 512B  mysqld                 0 4096B
mysqld                 0 48k   mysqld                 0 96k
sshd: root@           47B 144B  mysqld                 0 4096B
sshd: root@           47B 144B
sshd: root@           47B 144B
```

图 11-38 dstat 输出结果

11.4.6 列出挂载点和文件系统

Linux 系统中的每个分区都是用来组成整个文件系统的一部分,它采用了挂载点的处理方法,将一个分区和一个目录联系起来,载入的一个分区从一个目录下获得存储空间。文件系统挂载不正确、未关闭过多的文件操作或磁盘容量不足都会造成服务器响应过慢。下面介绍一些常用的命令,用于查看磁盘容量和文件系统信息。

(1) 使用 `mount` 命令查看挂载的文件目录,观察有没有挂载某个服务专用的文件系统,如 MySQL,有没有文件系统被重新挂载为只读模式,如图 11-39 所示。

```
[root@localhost ~]# mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,size=932796k,nr_inodes=233199,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

图 11-39 用 mount 命令查看挂载文件

(2) 使用如下命令查看挂载文件系统,观察文件系统的挂载选项是 `noatime` 还是 `default`,实例如图 11-40 所示。

```
cat /etc/fstab
```

```
[root@localhost ~]# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Fri Dec 4 13:33:57 2015
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root / xfs defaults 0 0
UUID=9482b2f7-2931-4323-9243-20ad7310dc21 /boot xfs defaults 0 0
/dev/mapper/centos-home /home xfs defaults 0 0
/dev/mapper/centos-swap swap swap defaults 0 0
```

图 11-40 fstab 查看挂载文件

(3) 使用命令 `vgs` 和 `lvs` 可以查找系统中存在的逻辑组和逻辑卷的信息列表，包括各个卷的大小和空闲容量，实例如图 11-41 所示。

```
[root@localhost ~]# vgs
VG      #PV #LV #SN Attr   VSize VFree
centos   1   3   0 wz--n- 59.51g 64.00m
[root@localhost ~]# lvs
LV      VG      Attr   LSize Pool Origin Data%  Meta%   Move Log Cpy%Sync Convert
home    centos -wi-ao---- 18.84g
root    centos -wi-ao---- 38.60g
swap    centos -wi-ao----  2.00g
```

图 11-41 vgs 查看卷组

(4) 使用 `pvs` 命令输出格式化的物理卷信息报表，包括卷大小和空闲容量，实例如图 11-42 所示。

```
[root@localhost ~]# pvs
PV      VG      Fmt  Attr PSize  PFree
/dev/sda2 centos lvm2 a--  59.51g 64.00m
```

图 11-42 pvs 输出结果

(5) 使用 “`df -h`” 命令查看磁盘容量的使用情况，输出结果如图 11-43 所示。从结果中，可以看出磁盘空间使用率为 36%（取自 `Use%` 列总和：7%+4%+1%+24%）。是否有大文件被删除但没有清空？如果磁盘空间有问题，是否还有空间来扩展一个分区？

```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/centos-root 39G  2.7G   36G   7% /
devtmpfs        911M   0   911M   0% /dev
tmpfs           921M   0   921M   0% /dev/shm
tmpfs           921M  33M   888M   4% /run
tmpfs           921M   0   921M   0% /sys/fs/cgroup
/dev/mapper/centos-home 19G   33M   19G    1% /home
/dev/sda1       497M  120M   378M  24% /boot
```

图 11-43 “df-h” 输出结果

(6) 使用 `lsdf` 命令列出当前系统打开的文件及目录，其中提供了大量关于这个应用程序本身与操作系统交互的信息。实例如图 11-44 所示，命令使用了 `+D` 参数，因此对应目录下的所有子目录和文件都会被列出。

```
[root@localhost ~]# lsdf +D /bin/
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF      NODE NAME
tuned    590  root   txt  REG  253,0    7136 134409173 /usr/bin/python2.7
dbus-daem 594  dbus   txt  REG  253,0   441256 134500997 /usr/bin/dbus-daemon
mysqld_sa 893  mysql  txt  REG  253,0   960384 134365792 /usr/bin/bash
mysqld_sa 893  mysql  255r REG  253,0   24723 134819500 /usr/bin/mysqld_safe
bash     5873 root   txt  REG  253,0   960384 134365792 /usr/bin/bash
```

图 11-44 lsdf 输出结果

11.4.7 过滤内核和中断信息

查看内核和中断信息，可以判断 swap 交换是否合理，TCP 连接怎么样，是否有异常系统日志。

1. 中断信息

(1) 使用 `sysctl` 命令可以显示 `/proc/sys` 目录中的内核参数，命令如下：

```
sysctl -a | grep ...
```

也可以通过中断报告文件，查看中断发生的次数，命令如下：

```
cat /proc/interrupts
```

从前面的中断报告文件中可以看出中断请求是否均衡地分配给了 CPU 处理，是否有某个 CPU 因为大量的网络中断请求或者 RAID 请求而过载。

(2) 查看 `ip_conntrack`

`ip_conntrack` 是连接跟踪数据库（conntrack database），代表 NAT 机器跟踪连接的数量，通过查看 `ip_conntrack` 来查找异常 IP，只要打开 `iptables` 服务就会开始跟踪。注意参数 `conntrack_max` 是否足够大，是否能够应付服务器的流量，`conntrack_max` 来自于 `/proc/sys/net/ipv4/netfilter/ip_conntrack_max`：

```
cat /proc/net/ip_conntrack
```

2. 内核消息

`dmesg` 命令可以用来显示内核环缓冲区（kernel-ring buffer）的内容，可以获得诸如系统架构、CPU、内存和挂载的硬件等多个运行级别的系统信息。操作系统启动时，内核将各种消息存放在这里，对于诊断系统问题非常有用。由于 `dmesg` 日志输出不能在一页中完全显示，因此结合使用管道（pipe）命令，将其输出到 `more` 或者 `less` 命令单页显示，示范命令如下，也可以使用 `tail` 或者 `grep` 等文字处理工具来处理：

```
# dmesg | more
```

不管是启动时还是系统运行过程中，只要是内核产生的信息，都会被记录到内存中的某个保护区段。如下实例可以搜寻启动时硬盘的相关格式。

质量全面管控：从项目管理到容灾测试

```
[root@www ~]# dmesg | grep -i hd
ide0: BM-DMA at 0xd800-0xd807, BIOS settings: hda:DMA, hdb:DMA
ide1: BM-DMA at 0xd808-0xd80f, BIOS settings: hdc:pio, hdd:pio
hda: IC35L040AVER07-0, ATA DISK drive
hdb: ASUS DRW-2014S1, ATAPI CD/DVD-ROM drive
hda: max request size: 128KiB
```

3. 操作系统日志

(1) 使用如下命令查看整体系统信息，包含系统启动期间的日志，还包括 mail、cron、daemon、kernel 和 auth 等内容：

```
less /var/log/messages
```

(2) 使用如下命令查看包含验证和授权方面的信息，如 POP3、SSH、Telnet、FTP 等：

```
less /var/log/secure
```

这两个日志中，需要关注以下两点：

- 查看错误和警告消息，是否是因为连接数过多所导致的。
- 是否有硬件错误或文件系统错误。

11.4.8 定时任务

crontab 服务是定时任务，在操作系统服务器的后台定时运行，在一定程度上占用 CPU 和内存，会影响整个服务器的性能。

(1) 查看 crontab 服务是否已经运行，命令如下：

```
ls /etc/cron* +cat
ps -ax | grep cron
```

(2) 查看各用户的定时任务列表，命令如下：

```
for user in $(cat /etc/passwd | cut -f1 -d:); do crontab -l -u $user; done
```

运行以上两个命令可以查看哪些用户提交了定时任务，现在是否仍有任务在执行，输出结果如图 11-45 所示。

```

[root@localhost ~]# ls /etc/cron* & cat
ls: cannot access *: No such file or directory
ls: cannot access cat: No such file or directory
/etc/cron.daily:
0hourly
/etc/cron.daily:
0yum-daily.cron logrotate man-db.cron
/etc/cron.hourly:
0anaconda 0yum-hourly.cron
/etc/cron.monthly:
/etc/cron.weekly:
[root@localhost ~]# for user in $(cat /etc/passwd | cut -f1 -d:); do crontab -l -u $user; done
no crontab for root
no crontab for bin
no crontab for daemon
no crontab for adm
no crontab for lp
no crontab for sync
no crontab for shutdown
no crontab for halt
no crontab for mail
no crontab for operator
no crontab for games
no crontab for ftp
no crontab for nobody
no crontab for avahi-autoipd
no crontab for dbus
no crontab for polkitd
no crontab for tss

```

图 11-45 查看定时任务

11.4.9 分析系统日志

在测试环境中，要关注那些关键的系统日志问题，以下为部分应用日志。

- Apache & Nginx: Web 服务器可以查看访问日志 (access.log) 和错误日志 (error.log)，直接查找 404、500、505 等典型 HTTP 错误代码，查看是否有 limit_zone 错误。
- MySQL & Oracle: 在数据库服务器上，查看安装目录下的 mysql.log 或 alert_<SID>.log，是否有错误消息，是否有结构损坏的表。
- Tomcat: 查看 Tomcat 目录下的 catalina.out 日志，是否有错误信息或数据库连接等异常信息。

11.5 要点回顾

本章讲述了如何使用 Zabbix 监控服务器和网络设备，使用 Grafana 展示 Zabbix 的监控数据，以及学会怎样整合 Zabbix 和 Grafana。另外，在遭遇服务器故障时，能够遵循流程排查原因。

- 查明在服务器上运行的都有哪些程序和进程。
- 明确是否和 I/O、硬件、网络或系统配置相关。
- 查看是否有熟悉的一些特征？比如对数据库索引使用不当，或者太多的 Apache 后台进程。这样有可能找到真正的故障源头。

第 12 章

灾难恢复与容灾测试

软件行业高速发展，软件测试专业同样也在不断成长。在 2005 年之前，测试同行主要的工作就是功能测试，也就是最基本的验证工作。作为软件研发过程的最后一道关卡，既重视又被轻视，因为上线时间已定，只能等待开发人员编码完成，才能进行功能测试。被重视表现在验证功能是否实现，被轻视表现在压缩的时间都是软件测试的时间。最极端的情况就是产品已经上线，但是仍然存在大量 Bug，上线之后再不断修改。这是因为软件测试的技术含量确实比较低，毋庸置疑。这是当时的现状。

后来，随着自动化测试工具的出现，稍微提高了测试人员的技术水平，有些具有研发经验的测试同行开始进行自动化测试，开始是录屏工具，例如 QuicktestPro，接下来是数据驱动的测试工具，例如 TestComplete，最后很多资深的测试同行开始白盒测试，使用 JUnit。这个阶段测试同行的待遇和水平都提升很多，拥有一定话语权，在评审会议上，开始注重软件测试的意见。

在 2008 年左右，因为用户使用量的大幅度提升，更多功能特性（例如秒杀等）成为常用功能，所以性能测试成为软件测试的主要类型，性能测试工具 LoadRunner 和 JMeter 开始作为性能测试的主要工具。

经过几年的经验累积和技术水平提升，测试同行对于功能测试、自动化测试、性能测试已经非常熟练。很多公司的产品经过严格的测试，正常运行已经成为常态。接下来就是管理和运维，确保软件在任何时间都不会影响用户的使用，那么容灾和灾难恢复更需要学习和掌握运用到实践工作中的测试类型。这是软件测试人员面临的新技术和新挑战。因为灾难恢复比较容易理解，本章先从灾难恢复开始讲解，再逐步地过渡到容灾测试。

12.1 灾难恢复

读者在了解灾难恢复的概念之前，有必要先了解灾难的定义。从一个软件 and 平台的角度讲，一切引起系统非正常停机的事件都可以称为灾难。

灾难大致可以分为以下 3 个类型：

- 自然灾害，就是人力不可抗拒的，包括洪水、地震、火灾等，这种灾难破坏性大，影响面广；
- 设备故障，就是硬件故障，包括主机的 CPU、硬盘等损坏，电源中断以及网络故障等，这类灾难影响范围比较适中，恢复手段就是更换硬件；
- 人为操作破坏，包括误操作、人为蓄意破坏等，例如著名网站因为系统人员误删除文件，导致网站 10 个小时无法访问。

在遇到灾难的时候，任何企业的软件和平台都会显得有些“弱不禁风”。例如 2016 年 5 月 27 日下午，大批网友反映，支付宝出现网络故障，无法登录、转账和付款。与此同时，打开余额宝后，连余额也不能显示，网络无法连接。傍晚 6 点左右，支付宝无线支付官方微博发布事故原由，并让用户放宽心，资金安全不会受此影响，原因是：“由于杭州市萧山区某地光纤被挖断，造成目前少部分用户无法使用支付宝，运营商正在抢修，支付宝工程师正在紧急将用户请求切换至其他机房，受影响的用户正在逐步恢复”。这是灾难恢复的最典型案例。属于第二个类型的灾难，典型的设备故障，该事件范围最广，影响最大。

通过上述实例，读者可以很容易得到灾难恢复的基本定义。灾难恢复测试就是灾难发生后，将生产平台恢复到正常运作的的能力。

12.1.1 灾难恢复的规范

随着国内社会信息化进程的全面加快，网络和信息平台的基础性、全局性作用日益增强，经济和社会发展对基础信息网络和重要信息系统的依赖越来越大，由此而产生的信息安全问题日趋严重。

2003 年 8 月，中央办公厅、国务院办公厅联合下发了《国家信息化领导小组关于加强信息安全保障工作的意见》，对基础信息网络和重要信息系统灾难备份与恢复做了原则规

质量全面管控：从项目管理到容灾测试

定，第一次提到了重要信息系统需要具备灾难恢复能力。

2004 年 9 月，国务院信息化工作办公室下发了《关于加强国家重要信息系统灾难备份工作的意见》，要求重要信息系统的各主管部门要制定行业的灾难备份建设政策和规划，对本行业各单位的灾难备份建设进行指导和管理，确保国家和行业灾难备份政策法规的贯彻落实；要求各重要信息管理运行机构要明确灾难备份管理和执行部门，负责落实国家和行业主管部门的灾难备份建设政策和要求，确定本单位的灾难备份建设目标和建设模式，制定完善的灾难恢复计划，确保在发生灾难和出现重大事故后能快速地恢复业务系统的运行。

2005 年 4 月，国务院信息化办公室联合银行、电力、民航、铁路、证券等 8 大重点行业，制定发布了《重要信息系统灾难恢复指南》，对国内各行业的灾难备份与恢复工作的开展提供了指导。

2007 年 7 月，经过两年的实施以及广泛征求意见，《重要信息系统灾难恢复指南》经过修改完善后正式升级为国家标准 GB/T 20988-2007《信息系统灾难恢复规范》，并于 2007 年 11 月 1 日开始正式实施。这是中国灾难备份与恢复行业的第一个国家标准，是各行业进行灾备建设的重要参考性文件，具有重大意义。该指南和国家标准最大的意义在于：对灾难备份、灾难恢复相关术语进行了规范和梳理，指明了灾难恢复工作的流程，明确了灾难恢复的等级和相关要素，制订了灾难恢复工作的主要环节及各环节具体工作，其中包括灾难恢复的管理，需求的确定，策略的制订和实现，预案的制订、落实和管理，预案框架等。

《重要信息系统灾难恢复指南》及《信息系统灾难恢复规范》的出台为人民银行、银监会、证监会、保监会等金融监管机构起草、制定灾难恢复行业相关标准提供了充分的参考依据和方向指引。

银行业灾备标准率先响应，随着《信息系统灾难恢复规范》以及相关政策、标准的出台，银行业率先出台了相关的行业政策和标准。2006 年 4 月，中国人民银行颁布了《关于进一步加强银行业金融机构信息安全保障工作的指导意见》（银发[2006]123 号文），要求全国性大型银行，原则上应同时采用同城和异地灾难备份和恢复策略；区域性银行可采用同城或异地灾难备份和恢复策略。2006 年 8 月，银监会发布的《银行业金融机构信息系统风险管理指引》（银监发[2006]63 号），明确地提出金融机构应制定信息系统应急预案，并定期演练、评审和修订，省域以下数据中心至少实现数据备份异地保存，省域数据中心至少实现异地数据实时备份，全国性数据中心实现异地灾备。

2008 年 2 月，中国人民银行正式发布和实施《银行业信息系统灾难恢复管理规范》

(JR/T0044-2008)。其中要求，短时间中断对国家、外部机构和社会产生重大影响或影响单位关键业务功能并造成重大经济损失的系统：RTO（恢复时间目标）<6 小时，RPO（恢复点目标）<15 分钟；短时间中断会影响单位部分关键业务功能并造成较大经济损失的系统：RTO<24 小时，RPO<120 分钟；短时间中断会影响单位非关键业务功能并造成较大一定经济损失的系统：RTO<7 天。

2009 年 6 月，为进一步加强商业银行信息科技风险管理，银监会发布了新的《商业银行信息科技风险管理指引》，原指引同时废止。新指引将信息科技治理作为首要内容提出，充实并细化了对商业银行在治理层面的具体要求；重点阐述了信息科技风险管理和内外部审计要求；对商业银行信息科技整个生命周期内的信息安全、业务连续性管理和外包等方面提出了高标准、高要求，使可操作性更强。

提示：以下是关于灾难恢复的重要的法律法规：

- 2003 年 8 月中国中办《国家信息化领导小组关于加强信息安全保障工作的意见》。
- 2004 年 9 月中国人民银行《关于加强银行数据集中安全工作的指导意见》。
- 2006 年 8 月银监会《银行业金融机构信息系统风险管理指引》此文件已废除，新文件《商业银行信息科技风险管理指引》于 2009 年 6 月发布。
- 2006 年 8 月证监会《证券公司集中交易安全管理技术指引》。
- 2008 年 3 月保监会《保险业信息系统灾难恢复管理指引》。

由于这个规范目前是灾难恢复测试的纲领性文件，那么就必须仔细研读，找出其中的重点。《银行业信息系统灾难恢复管理规范》规定了信息系统灾难恢复应遵循的基本要求，适用于信息系统灾难恢复的规划、审批、实施和运维。主要包括以下几部分内容：

- 灾难恢复行业相应的术语和定义。
- 灾难恢复概述（包括灾难恢复的工作范围、灾难恢复的组织机构、灾难恢复规划的管理、灾难恢复的外部协作、灾难恢复的审计和备案）。
- 灾难恢复需求的确定（包括风险分析、业务影响分析、确定灾难恢复目标）。
- 灾难恢复策略的制定（包括灾难恢复策略制定的要素、灾难恢复资源的获取方式、灾难恢复资源的要求）。
- 灾难恢复策略的实现（包括灾难备份系统技术方案的实现、灾难备份中心的选择和建设、专业技术支持能力的实现、运行维护管理能力的实现、灾难恢复预案的实现）。

质量全面管控：从项目管理到容灾测试

由此可见，《银行业信息系统灾难恢复管理规范》对灾难恢复建设的全流程实现给出了详细的指导意见，具有很高的可操作性。

《银行业信息系统灾难恢复管理规范》中对灾备恢复资源 7 要素的详细定义，可以引导 IT 行业公司全面考虑灾难恢复建设的几个重要方面，如表 12-1 所示。

表 12-1 灾备恢复资源 7 要素

序号	要素	要素的考虑要点
1	备用基础设施	灾难备份中心选址与建设； 备用的机房及工作辅助设施和生活设施
2	数据备份系统	数据备份范围与 RPO； 数据备份技术； 数据备份线路
3	备用数据处理系统	数据处理能力； 生产系统的兼容性要求； 平时的状态（处于就绪还是运行）
4	备用网络系统	备用网络通信设备系统与备用通信线路的选择； 备用通信线路的使用状况
5	灾难恢复预案	明确灾难恢复预案的要素： A）整体要求； B）制订过程的要求； C）教育、培训和演练要求； D）管理要求
6	运行维护管理能力	运行维护管理组织架构； 人员的数量和素质； 运行维护管理制度； 其他要求
7	技术支持能力	软件、硬件和网络等方面的技术支持要求； 技术支持的组织架构； 各类技术支持人员的数量和素质等

12.1.2 灾难恢复能力等级

灾难恢复等级的确定是信息系统灾备建设的重要考虑因素。《银行业信息系统灾难恢复管理规范》将灾难恢复能力划分为 6 级，如图 12-1 所示。

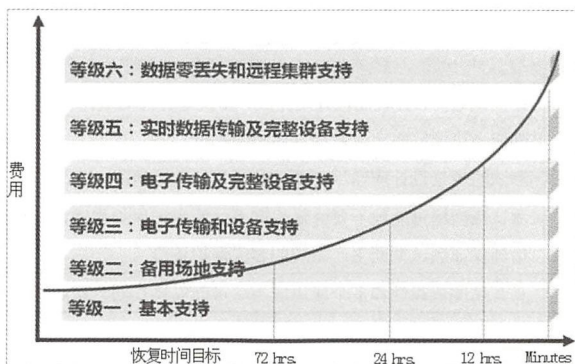


图 12-1 灾难恢复能力等级

从技术的角度来看，可以理解 6 个不同等级的区别。

- 1、2 等级要求企业对生产系统做备份，并且定期把备份数据通过一定的方式（如转运磁带）放到第二个站点。
- 3、4 等级与 1、2 等级最大的区别是，企业需要周期性地对数据进行备份，并通过电子链路和网络在线传输到灾备中心。
- 5、6 等级要求数据不是周期性的保护，而是实时地传输到备份站点，并要求在故障发生时，系统可以自动地切换到灾备站点。针对不同等级的数据保护要求。

依赖于常规的备份系统很难很好地完成零数据丢失保障。在较高的数据完整性要求基础上，再如电子支付又是一个几乎完全面向公众用户的服务，具有比较高的业务连续性要求，基本要求 7×24×365 天不间断运行，即使在节假日也要求电子支付平台正常运转。用户就餐大部分是使用信用卡付款，那么 POS 交易系统就是一个典型的绝对保障高数据完整性的业务系统。为了支持零数据丢失和业务连续性保障需要一个周全的业务连续性计划来加以支持。备份、灾难备份、持续数据保护和多样化恢复手段都应该成为核心业务连续性计划的必要组成部分来共同完成业务连续性计划目标。目标过多会导致有些混乱，笔者列出如表 12-2 的表格，读者可以作为参考。

表 12-2 灾难恢复的注意事项和原则

注意事项	原则
对业务运营的影响	应当尽量避免或降低对正常业务运营的影响
关键的业务周期	测试应当尽可能安排在非业务高峰期，以避免或降低风险
分离关键的组件	如果测试对特定业务的中断是无法避免的，那么应当在一个可接受的时间段内，将所涉及的 IT 架构组件与所有会受影响的业务部门隔离开，进行测试
保证足够的人员支持生产系统	完整的测试应当被分为多个可管理的批次，目的是保证有足够的资源维护和支持生产系统

续表

注意事项	原则
恢复流程准备就绪	在测试之前，管理流程开发完成，要恢复的构架组件和应用相关的灾难恢复方案与恢复步骤应当已经编写完成并经过适当的测试
对真实场景的模拟程度	测试的场景设计应当尽可能反映最坏的灾难情况
测试期间的容灾保护程度	测试应当尽可能地不要降低生产系统的容灾保护程序，如果无法避免，应当考虑采用一些措施保证测试期间生产系统的容灾保护备份
应用系统的分组	具有高度依赖性和多个应用应当尽可能安排在同一次测试中

12.1.3 灾难恢复的关键指标

目前 IT 行业公认最关键的衡量指标有两个：一个是 RTO，另一个是 RPO。

所谓 RTO (Recovery Time Objective) 是指灾难发生后，从软件系统宕机导致业务停顿时开始，到软件系统恢复至可以支持各部门的运作、恢复运营时，此两点之间的时间段称为 RTO。

所谓 RPO (Recovery Point Objective) 是指从系统和应用数据而言，要实现能够恢复到可以支持各部门业务运作，使系统及生产数据应恢复到怎样的更新程度。这种更新程度可以是上一个自然日的备份数据，也可以是上一次交易的实时数据。

图 12-2 展示了 RTO 和 RPO 的关键指标。

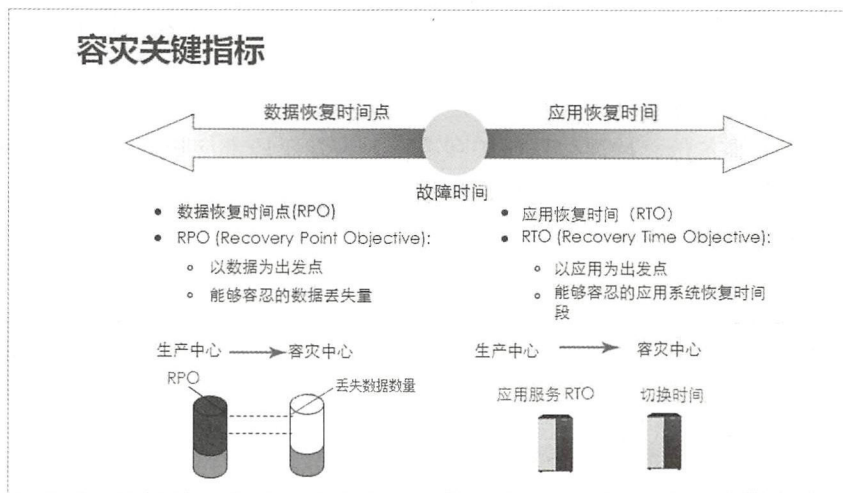


图 12-2 RTO 和 RPO

提示：RTO 关注点是业务恢复时间，RPO 关注点是损失的数据量。

下面详细说明 RTO 和 RPO，在《银行业信息系统灾难恢复管理规范》中，也给出了某个特定行业灾难恢复能力等级与 RTO、RPO 之间关系的示例，如表 12-3 所示，可作为参考。

表 12-3 灾难恢复能力等级与 RTO、RPO 之间关系

灾难恢复能力等级	RTO	RPO
1	2 天以上	1 天至 7 天
2	24 小时以后	1 天至 7 天
3	12 小时以上	数小时至 1 天
4	数小时至 2 天	数小时至 1 天
5	数分钟至 2 天	0 至 30 分钟
6	数分钟	0

RTO 是数据中心可容许服务中断的时间长度。比如说服务发生后半天内便需要恢复，RTO 数值就是 12 小时。RTO 是反映数据中心业务恢复的及时性指标，表示业务从中断到恢复正常所需的时间，RTO 数值越小，代表容灾系统的数据恢复能力越强，数据中心可以部署很多容灾系统，来获取最小的 RTO，但这意味着投入大量资金。

提升 RTO 的常用技术有：磁带恢复、人工迁移、应用系统远程切换，这几种技术的 RTO 的表现如表 12-4 所示。

表 12-4 提升 RTO 的常用技术

容灾技术	时长
磁带恢复	日级
人工迁移	小时级
应用系统远程切换	秒级

部署不同的容灾技术将获得不同的 RTO 值，从业务连续性角度考虑，肯定希望 RTO 数值越小越好，尤其是很多互联网数据中心，中断几分钟都会损失数百万的成交量，这些数据中心往往不惜一切代价要确保数据中心不中断运行。应用系统的自动切换涉及到数据中心网络、存、服务器等多方面的技术，不管数据中心任何一个位置出现了故障，这些部分都会启动软件系统进行切换，可以是设备之间的切换，也可能是集群之间的切换，还可能是异地数据中心切换，通过应用系统自动切换将业务转移到其他正常的系统中，然后再对故障设备进行排查。将故障原因找到并排除后，再将业务切回到原有系统中，应用系统



切换做得好，这个过程不会引起业务的二次中断，让用户无感知切换。

RPO 是数据中心能容忍的最大数据丢失量，也是当业务恢复后，恢复的数据所对应时间点。RPO 取决于数据中心的数据恢复到怎样的更新程度，这种更新程度可以是上一周的备份数据，也可以是昨天的数据，这和数据备份的频率有关，为了改进 RPO，必然要增加数据备份的频率才行。RPO 是反映数据中心恢复数据完整性的指标。在同步数据复制方式下，RPO 等于数据传输时延的时间，在异步数据复制方式下，RPO 基本为异步传输数据排队的时间。提升 RPO 的常用技术有：磁带备份、定期数据复制、异步数据复制、同步数据复制等，这几种技术的 RPO 的表现如表 12-5 所示。

表 12-5 提升 RPO 的常用技术

容灾技术	时长
磁带备份	日级
定期数据复制	小时级
异步数据复制	分钟级
同步数据复制	秒级

RPO 指标考验着数据中心数据复制能力，这并不意味单纯增加数据复制的频率，因为应用的高峰时段无法进行备份操作，而且备份数据本身所花费的时间也会过长，数据复制频率增加到一定程度反而会降低 RPO 时长。现在出现镜像技术和快照技术可以有效地改进 RPO，往往可以将 RPO 缩小到秒级。

RTO 和 RPO 指标对于数据中心非常关键和重要，RTO 主要考验数据中心发生故障时，业务切换到容灾系统或者备份系统的能力，RPO 主要考验数据中心数据备份能力，尤其是当数据中心发生故障时，仍要具备一定的数据备份能力。但数据中心也不能过分地追求 RTO 和 RPO，因为 RTO 和 RPO 越小，意味着投资将越大。而总体投入成本越高，投资回报率将越低，从经济角度考虑，最好的容灾解决方案不一定是效益最好的容灾方案，容灾方案的总体投入和投资回报也是必须要考虑的设计指标，最佳的解决方案必须是在 RTO、RPO、运维及价钱多方面，都能够达到平衡。所以要理性看待 RTO 和 RPO，一方面努力设计一些新的容灾技术，另一方面还要简化容灾技术的复杂度和造价，不要一味追求 RTO 和 RPO 指标。过度追求 RTO 和 RPO 指标，甚至做到两者都是零，反而让数据中心更加臃肿，运维难度加大，耗费资金过多，数据中心要避免陷入单纯追求提升两个指标的怪圈，结合数据中心实际情况，因地制宜地适当提升两个指标。



此外, NRO (Network Recovery Object, 网络恢复目标) 和 DOO (Degrade Operation Object, 降级运作目标) 对容灾系统也是至关重要的性能指标。

- NRO 代表灾难发生后, 网络切换需要的时间。
- DOO 代表的是恢复完成以后到第二次故障或灾难的所有保护恢复以前的时间间隔, 反映了系统发生故障后降级运行的能力。

12.2 容灾测试

在上一节读者了解灾难恢复的定义和灾难恢复的等级和指标, 那么接下来读者可以了解到什么是容灾测试、容灾的定义、容灾测试与功能测试、容错测试和备份的区别。

12.2.1 容灾的起源

阅览新闻, 网络购物, 电子支付, 甚至订餐购票, 这些人们日常生活都大量存在于互联网环境中, 任何关键信息平台 (例如支付宝、12306 网站)、如果出现运转中断或者数据丢失都将导致不可估量的损失。IDC 统计得到的数据是: 网上银行每分钟运转中断损失 7000 美元, 企业资源管理系统为 13000 美元, 而呼叫中心更是高达 27000 美元, 同时统计的另外一组数据显示, 美国在 2000 年以前的 10 年间, 发生过灾难的公司中, 有 55% 当时倒闭, 剩下的 45% 中, 因为数据丢失, 其中的 29% 也在两年之内倒闭, 生存下来的仅占 16%。

正是由于容灾如此重要, 在 2002 年 7 月 26 日, 美国国会通过了涉及会计职业监管、公司治理、证券市场监管等一揽子改革的《公众公司会计改革和投资者保护法》(也称《萨班斯法案》)(Sarbanes-Oxley 法案, 简称 SOx 法案)。其中第 404 条款更因其严厉性和高昂的执行成本而闻名, 包含了基于 IT 高业务连续性的系统高要求。赴美上市前提是必须符合 SOx 法案的要求。针对于公司治理的 SOx 法案, 虽然主要指向财务管理、经营管理, 要求企业保证财务数据的真实性、全面性和及时性, 但对于境外融资的企业来说, 财务、帐务、生产等系统, 都是依赖于 IT 系统的。所有 SOx 法案规则的落脚点, 恰恰是 IT 系统的高业务连续性的体现, 因此, 建立容灾系统也是赴美上市的前提条件之一, 这是目前一般企业高层关注容灾测试的关键所在, 也是必须要学习和掌握的根本所在。

提示: 一定要区分下面两个 IDC。



一个是互联网数据中心（IDC），国内大型的数据中心有联通（北方较多），电信（南方较多），简单理解就是公司的服务器托管中心。

另一个 IDC 是国际数据集团（IDG）的全资子公司，其母公司 IDG 创建于 1964 年，公司致力于提供科技信息，提供资讯和分析资源。IDC 是全球领先研究咨询公司，著名的信息技术、电信行业和消费科技咨询、顾问和活动服务专业提供商。顾名思义就是提供数据最权威的公司。

12.2.2 容灾的定义

IT 业务系统或者 IT 服务平台，是互联网公司从事多项业务的基石，而存储在数据库中的数据更是关系到公司的生存和发展，在大数据时代的今天，一切都是数据为王，那么数据完整性是整个公司业务继续运行的基础性工作也是最重要的工作。

据此读者可以得到容灾的基本定义，容灾就是当各种灾难发生时，在保证生产系统的数据尽量少丢失的情况下，保持生产系统的业务不间断地运行。任何针对可能发生的灾难，提高系统的可靠性和可用性的努力都可以称为容灾。

容灾的目的就是在灾难发生时减少数据丢失和计划外宕机时间，保证业务平台可以连续性的运行。

如图 12-3 所示是 IT 业务系统发生各种灾难的比例分配表。

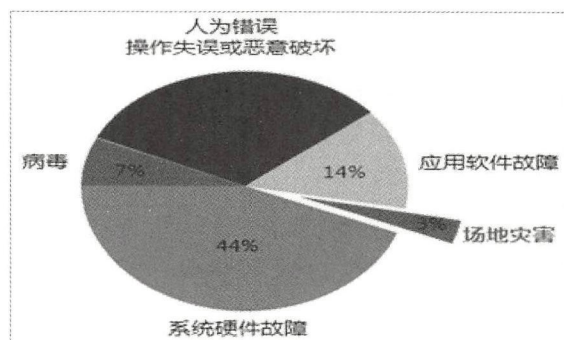


图 12-3 IT 业务系统发生各种灾难的比例分配表

容灾根据对系统的保护程度还可以进行精确划分，可以将容灾分为三部分：硬件容灾、数据容灾和业务容灾。





(1) 硬件容灾比较容易理解,在自然灾害,设备故障,人为破坏的三种典型灾难中任何一种灾难发生时,在保证生产系统的数据能尽量少的丢失的情况下,保持生产系统的主要业务不间断运行。

硬件容灾关注重点如自然灾害、设备故障、意外断电和人为误操作,很多情况是可以进行模拟测试。

(2) 数据容灾是指建立一个异地的数据备份和恢复系统,该系统是关键应用数据的一个实时复制与备份。

数据容灾是三个容灾类型中最重要的,因为硬件出现问题,可以重新购买,部署的应用程序出现问题可以重新搭建,而数据出现问题,一般情况下是很难恢复。所以非常有必要建立数据库灾备中心,与主库实时进行数据同步,做到实时备份,这也是最难的地方。

(3) 应用容灾是在数据容灾的基础上,在异地建立一套完整的与本地生产系统相同的备份应用系统,可以是互为备份。在灾难突发情况下,远程系统迅速接管并保证平台业务正常运行。

应用容灾是容灾系统建设的目标,也是三个容灾类型中花费时间最长的,例如,目前流行的支付系统,部署的应用非常多,那么需要做各个方面的应用容灾测试,例如消息中间件、收单系统、清结算系统、虚拟银行通道等。如图 12-4 是一个系统容灾建设简图。

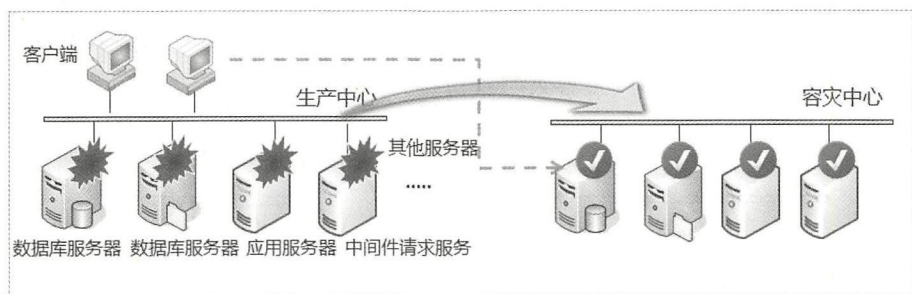


图 12-4 系统容灾建设简图

因为容灾是极其重要的,所以一般是由专业的运维团队一直在持续关注,并且高效地自动化管理。容灾主要目的是为了预防一些不可预料的意外,比如火灾、地震、紧急硬件故障等。成熟的公司大约要一个月进行一次容灾测试。

提示:运维团队的主要工作包括监控平台运行情况,硬件指标有无异常,常用的应用使用统计,系统升级,软件更新,数据库备份,网络维护,新系统上线,商业软件打补丁。





12.2.3 容灾的区别

1. 容灾与功能测试

功能测试的本质是验证单个功能是否正常运行，可以实现需求规格说明书的要求。例如一个简单的财务系统，一级功能模块是信息管理，二级模块包括个人信息、基本权限和组管理。执行的功能测试用例是填写个人信息录入财务系统，其测试结果是添加成功或者添加失败，还有可能出现的 Bug 是填写姓名或者学校的时候，输入超长汉字如 20 个，页面提示错误。

容灾测试的本质是灾难发生时，尽量减少数据丢失和计划外宕机时间，保证业务平台可以连续性的运行。例如使用自动化脚本多次循环填写个人信息（姓名从 tom1 到 tom10），在第一个自动化脚本 tom1 开始执行的时候，关闭主数据库服务器，切换到备份服务器，等待 10 个自动化脚本全部执行完成，查看测试结果，统计出 RTO 和 RPO 数值，例如 RTO 是查看从主数据库切换到备份数据库花费的时间，RPO 是几个用户信息录入成功，多少个失败。

2. 容灾与备份

一般意义上，容灾指的是不在同一机房的数据或应用系统备份，备份指的是本地的数据或系统备份。通常说的灾备是将容灾与备份结合，即本地备份结合远程数据复制实现完善的数据保护。如图 12-5 所示就是容灾与备份结合的实例图。

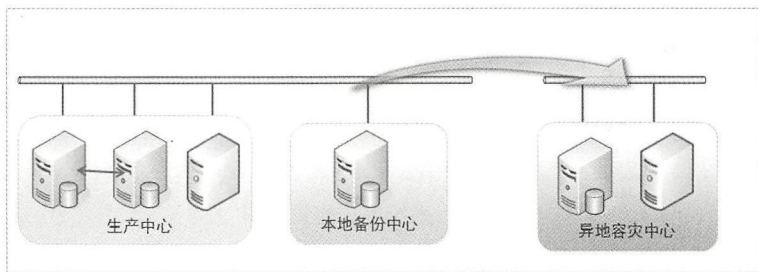


图 12-5 容灾与备份

3. 容灾与容错

在清楚容灾的基本定义后，还需要了解另外一个重要的测试类型“容错测试”（Fault Tolerance），这是最容易和容灾测试混淆。





容错测试主要检查系统或平台的容错能力，检查软件在异常条件下自身是否具有自动恢复性的措施或者某种灾难性恢复的手段。已经完成容错测试的系统或平台能确保系统不发生无法意料事故。

例如：公司平台的用户不断增加，导致服务器上的收单业务经常宕掉，此功能模块是使用 Tomcat 作为应用服务器，系统人员不可能 24 小时一直查看服务器状态，这时就需要一个自动监测 Tomcat 并且出现问题还可以重启 Tomcat 的脚本，它的工作原理是首先检测 Tomcat 进程是否存在，如果不存在就启动，如果进程存在，检测页面返回码状态，如果是 200 就是正常，如果不是就重启 Tomcat 服务器。

容错简单理解就是重要功能模块是否具有出现错误时可以自我恢复的能力。

所以容错和容灾最大的区别是，容灾必须通过系统冗余、灾难检测和系统迁移等技术来实现。当设备故障不能通过容错机制解决而导致系统宕机时，这种故障的解决就属于容灾的范畴。

4. 容灾与灾难恢复

要区分容灾与灾难恢复的不同，读者可以首先根据两者的定义进行对比。

- 容灾 (Disaster Tolerance)，就是在灾难发生时，在保证生产系统的数据尽量少丢失的情况下，保持生存系统的业务不间断地运行。
- 灾难恢复 (Disaster Recovery) 就是在灾难发生后，将系统恢复到正常运作的能力。

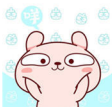
至此，读者应该非常清楚容灾与灾难恢复的本质区别。容灾强调的是在灾难发生时，保证系统业务持续不间断运行的能力。而灾难恢复强调的是灾难之后，系统的恢复能力。

12.3 详解容灾测试

上述小节主要讲述了容灾的起源和基本定义，并且详细说明容灾与功能测试、容灾与备份、容灾与容错、容灾与灾难恢复的区别。读者有了容灾的基础知识，那么接下来就要了解容灾的目标，执行容灾测试的参与者，重点是容灾测试的流程。

12.3.1 容灾测试的目标

容灾项目具有以下的一些特性：





质量全面管控：从项目管理到容灾测试

- 项目需求无法提前计划，所有容灾的设计都是在持续优化和测试过程中不断发现新的需求。
- 容灾测试点相互交错复杂，并行测试难度高，测试效率低。
- 项目测试环境依赖多，复杂度高。
- 项目研发涉及到的业务场景覆盖广，涉及人员多，测试数据准备工作量大。
- 测试难度较其他普通业务项目要求高，需要同步根据日志定位所有的测试场景出现原因。

通过上述内容读者能够清楚容灾项目具有非常多的难点，所以在进行测试前必须要定下容灾的基本目标，一定要遵循以下 4 个标准：

- (1) 模拟极端错误发生，测试业务恢复功能和业务持续性流程。
- (2) 发现平台潜在隐患，确保出现突发情况时平台能够正常运行。
- (3) 在极端访问量的情况下，牺牲一小部分非主要业务功能或者一小部分用户体验，保障整体系统的稳定以及主要功能的正常运行。这就是分流，淘宝、京东都有这样的容灾机制。

(4) 进行容灾测试时，需要同步分析日志。因为容灾一般都是测试的异常场景，测试环境会经常造成人为的不稳定，出现各种异常，所以测试时一定要仔细结合日志分析，确认当前展示的结果是否是因为容灾测试用例生效而出现的。

12.3.2 职责的划分

- (1) 测试工程师
 - 分析业务影响。
 - 识别关键业务功能和应用系统。
 - 识别应用系统之间的相互关联和支持关系。
 - 确定在预定的时间内无法正常运行时，对关键业务造成的损失及影响（定性和定量）。
 - 确定灾难恢复和业务连续需求（RTO 和 RPO）。
 - 识别关键的服务时间段和可容忍的性能下降程度。
- (2) DBA
 - 随时查询容灾库的数据，保证数据完整性。





- 确保容灾库和生产库一直同步，查询得到的结果都是最新的数据，容灾库会随着生产中心数据的变化不断更新。
 - 充分挖掘容灾端设备的利用率，使投资收益最大。
- (3) 运维工程师
- 与开发工程师和测试工程师协商，准备监控哪些指标。
 - 部署 Zabbix 或者 Cacti 等监控平台，监控定义的指标，具体如图 12-6 所示。

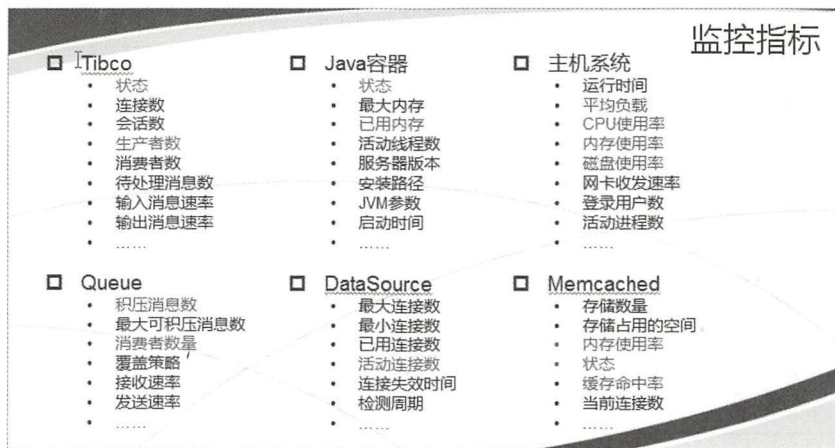


图 12-6 监控指标

12.3.3 容灾测试的流程

整个容灾测试的过程可以看做一个项目管理的过程。由于每次容灾测试都要调动公司大量资源，同时也会对业务造成影响，因此为了确保容灾测试成功或者及时回退，必须制定详细而周密的容灾测试计划和测试方案，经过各方讨论确定后才可实施。

在正式容灾测试之前，为了确保数据的安全，必须对业务数据进行备份；同时，为了确保测试脚本、流程和实际情况的一致，必须进行平台总体架构和功能流程检查；此外，容灾测试一般时间比较长，必须事先进行资源协调，做好人员的调配工作和支持的工作。

另外，容灾测试后必须进行灾难恢复流程的整理及改进，形成测试总结报告，为以后的容灾测试提供经验教训。

可以通过多种不同的形式进行测试，一般不建议直接进行整体测试，在这之前，通过不同级别的功能模块循序渐进地进行测试，从而减少对业务的影响并不断完善容灾测试流程。



质量全面管控：从项目管理到容灾测试

容灾测试的执行列表，如表 12-6 所示。

表 12-6 容灾测试的执行列表

编号	要求	描述
1	灾难切换时间要求	灾难系统切换时间不超过 30 分钟，在 10 分钟以内实现
2	多种灾难切换方式	提供自动灾难系统切换和手动灾难切换方式
3	计划内维护要求	提供计划内维护支持能力，计划内维护切换时间不多于 10 分钟
4	数据丢失性要求	原则上要求零数据丢失，可以依据情况进行调整
5	数据同步方式	提供同步和异步两种方式
6	备份和灾难备份方式	采用物理备份方式实现
7	生产系统的性能影响要求	生产系统性能影响不超过 10%
8	生产系统可用性要求	容灾系统不会降低生产系统可用性
9	网络链路容错	支持网络链路的容错，可以利用网络的备份链路，比如多路网卡等
10	灾难系统的硬件故障	由于灾难系统硬件故障导致的灾难系统不可用，所以不会对生产系统产生影响，比如网卡，磁盘以及控制卡等
11	灾难系统的软件故障	由于灾难系统软件故障导致的灾难系统不可用，所以不会对生产系统产生影响，比如灾难系统管理软件
12	在线实施要求	要求在备份系统实施期间保持生产系统运行
13	容灾切换要求	提供一对多和多对一的容灾复制和切换
14	一键切换	支持单键切换，所有切换都要求提供字符菜单、命令行和图形界面 3 种模式，所有切换都要求一步完成
15	全业务切换	支持业务系统级别切换，进行完整的 Oracle 数据库，Tibco 中间件，Web 服务器及其他核心部件的自动切换
16	业务部件切换	支持业务部件级别切换，在业务系统某一部件发生灾难时，切换该部件到容灾系统中
17	网络切换	指定在切换时候是否支持网络切换

12.4 容灾测试实战

容灾测试的要点可归纳为“1、2、3、4”：

- (1) “1”代表一个核心原则，即基于业务影响分析，全面提高 IT 系统的抗风险能力。
- (2) “2”代表要关注 RTO（恢复时间）和 RPO（数据丢失量）两个重要指标。
- (3) “3”代表要做好三件事，包括数据传输、业务切换和容灾演练与监控。





(4) “4”代表实现操作系统、文件、数据库和应用四项恢复。

本次实例的内容因为篇幅所限，仅列举常用的数据库和中间件 Tibco 以及移动支付的主流程。

容灾测试的难点在于前期需要梳理应用与中间件、缓存、存储、公共服务等的依赖关系，测试的策略是模拟极端错误发生，测试业务恢复功能和业务持续性流程，目标是发现平台潜在隐患，确保出现突发情况平台能够正常运行。所以执行容灾测试难度非常大，要求非常高，需要配合的部门非常广。

12.4.1 容灾测试计划

目前互联网公司进行的容灾测试，不仅仅是单独的执行容灾，还包括灾难恢复、备份、容错。上文中笔者详细描述了这些测试类型的定义，并且给以实例进行了区分。因为本书进行的容灾测试包括了上述所有的类型。如表 12-7 首先需要列出一份详细的工作计划。

(1) 容灾测试时间表，具体工作，执行人员都必须定义清楚。

表 12-7 容灾测试时间表

工作元素	开始日期/结束日期	执行人员	注释
测试方案	1.1-1.3	Tom	说明测试项目的具体方案
测试方案评审	1.3-1.6	Tom	先内部评审，然后项目组成员评审
用例设计	1.7-1.11	Tom	编写测试用例
测试用例评审	1.12-1.14	Tom	先内部评审，然后项目组成员评审
测试执行	1.14-1.19	Tom	执行容灾测试用例
测试分析	1.20-1.20	Tom	每周周会汇报的测试情况
回归测试	1.21-1.28	Tom	开发人员修改 Bug 完成，开始第二轮容灾测试
测试报告	1.28-1.30	Tom	也叫测试总结

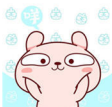
(2) 容灾测试目标

总体目标：不仅保护数据，更重要的目的是保证业务的连续性。

容灾目标：主要针对数据库、Tibco 中间件和移动支付应用程序正常运行，还有突发的异常情况。查看数据库、Tibco 和移动支付应用程序各自的 RTO 和 RPO。

备份目标：数据库实时备份验证。

容错目标：查看 Apache 和 Tomcat 是否实现自动重启。





质量全面管控：从项目管理到容灾测试

灾难恢复目标：验证硬件服务器主备。

（3）列出优先级

见表 12-8。

表 12-8 容灾测试优先级

级别	具体测试内容	测试执行时间
最高优先级	Tibco, 数据库	测试时间 3 个工作日
中度优先级	移动支付	测试时间 2 个工作日
普通优先级	Apache, Tomcat	测试时间 1 个工作日

（4）测试环境搭建

容灾测试必须重新搭建一套新的测试环境，标准是和系统测试一模一样的环境，原因是容灾测试需要经常停止服务器或者经常断开应用服务，如果和系统测试共用一套环境，影响各自工作，关键是无法定位问题原因，是功能没有实现，还是因为应用关闭导致的问题。

（5）测试数据准备

容灾测试必须使用自动化脚本，原因本书已经在容灾测试与功能测试时进行了详细说明。如果不使用自动化脚本，不同的容灾场景无法连续执行，需要手工在相同的时间内不同的机器上执行不同的场景，这样实现难度较大，就需要投入更多的人力，这是应该极力避免的。

数据库的数据一定要先进行统计，得到每张业务表的数值，这样可以准确得到 RPO 的数据。

（6）容灾测试总结

在容灾测试过程中一定要详细记录，包括操作方式、实现过程和测试用例执行结果，为后期执行容灾测试做参考。

12.4.2 容灾用例与 Bug

Bug 的书写规范和实例已经在第 5 章详细讲解，并且有相关实例，读者可以参考。本小节容灾用例与 Bug 书写在一起是便于读者的查看。为使读者更清楚用例编写的过程，在每个容灾测试用例之前笔者都有附图和相关文字说明。第一个移动支付主流程如图 12-7 所示。

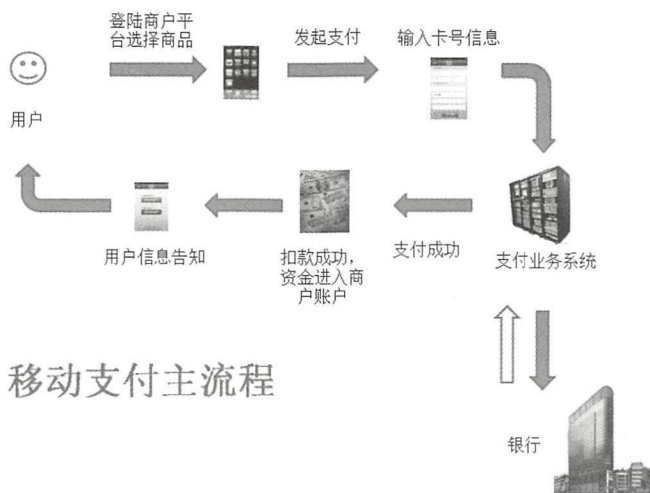


图 12-7 移动支付主流程

(1) 见表 12-9。容灾测试用例——移动支付主流程

表 12-9 容灾测试用例

测试用例编号和名称	0001-支付业务系统与银行数据传输
测试目的	测试支付业务系统的容灾情况
前提条件	1. 测试人员要编写一套 mock 系统即银行接口模拟器，便于制造用户数据； 2. 银行接口模拟器的工作原理是支付业务系统发送报文，模拟器接收并分发报文，再进行 XML 解析，最后将报文传回给支付业务系统。通常情况下返回成功和失败两个状态
测试步骤	1. 用户登录 mock 系统，输入测试用户的数据； 2. 通过手机端测试进行支付测试； 3. 输入各种银行卡信息，发送到支付业务系统； 4. 支付业务系统发送报文给银行模拟器，返回值给支付业务系统； 在传输给银行模拟器过程中，关闭主支付业务系统
预期结果	在页面显示支付失败，请重新输入信息
实际测试结果	不出现提示，页面显示 HTTP Status 404
关键点	一般公司肯定会有两套系统作为主备，如果出现特大的问题。例如本次测试的系统宕机情况，那么必须及时切换。详细记录 RTO 和 RPO 两个指标的数据

(2) 容灾测试用例——Tibco 中间件

Tibco EMS 是一款基于 JMS 标准的消息中间件产品，具有良好的扩展性、易用性和高效性，很多公司将其作为企业内部的消息总线，在企业应用集成平台架构中担任重要的角色。



质量全面管控：从项目管理到容灾测试

其特点是：提供可靠传输服务，分布式队列实现一对多信息传送；支持多种通信协议，适应不同的网络环境；负载均衡，根据可用资源动态地分配工作负载。

正因为 Tibco 的重要性，所以本次容灾测试选择了 Tibco 作为一个测试项。

表 12-10 是测试 Tibco 的几个状态，据此设计测试用例，由于篇幅所限，本次测试选择了最关键的 Down（进程宕掉）状态作为实例讲解。

表 12-10 Tibco 的测试项

序号	测试重点	现状	处置机制	建议处理方法
1	Tibco 状态	已有监控	记录日志并发送短信告警	自动重启并发送短信告警
2	Pending Message 监控	已有监控	发送短信告警	发送短信告警并重启对应的应用列表
3	hang-up（进程挂起）	已有监控	发送短信告警	要求重启对应的 Tibco 服务，并邮件发送，如果在一个小时内这台机器重启大于一定数次，则不再重启，直接发送邮件给相关人员
4	Down（进程宕掉）	已有监控	发送短信告警	同上
5	进行 Kill、Stop 等操作	已有监控	发生短信告警	同上

表 12-11 是具体测试用例设计。

表 12-11 Tibco 测试用例设计

测试用例编号和名称	0002—Tibco down 状态
测试目的	查看 Tibco
前提条件	采用主备模式部署
测试步骤	1. 进行支付业务过程； 2. 关闭 Tibco 主机
预期结果	正确切换到 Tibco 备份机
实际测试结果	提示支付业务失败 log，没有实现主备自动切换
备注	Tibco 服务日志显示 Tibco 主备服务器之间心跳停止，Tibco 服务器自动执行主备切换，从 Tibco 监控信息可以分析出 Tibco 服务器虽然自动回复，但是应用一直联系在原来的主机上，并没有完全切换成功，这种情况下，恢复 Tibco 服务必须执行 kill -9（假死 Tibco 的主进程），并且重新启动



(3) 备份测试用例——数据库

MySQL 支持单向、异步复制，复制过程中一个服务器充当主服务器，而一个或多个其他服务器充当从服务器。主服务器将更新写入二进制日志文件，并维护日志文件的一个索引以跟踪日志循环。

当一个服务器连接到主服务器时，它通知主服务器从服务器在日志中读取的最后一次成功更新的位置。从服务器接收时起发生的任何更新，然后封锁并等待主服务器通知下一次更新。

在实际项目中，主机和备份机上安装有 MySQL 数据库，两台服务器互为主备，要求当其中一台机器出现故障时，另外一台能够接管服务器上的应用，这就需要两台数据库的数据要实时保持一致，在本实例中使用 MySQL 的同步功能实现双机的同步复制。见表 12-12。

表 12-12 MySQL 备份测试用例

测试用例编号和名称	0003-数据库同步测试
测试目的	验证数据库数据是否同步
前提条件	手工执行数据库同步
测试步骤	1. 配置数据库后进行测试，首先在网络正常情况下测试，在 A 库上进行数据库操作，和在 B 库上进行数据库操作，数据都能够同步。 2. 拔掉 B 库的网线，然后在 A 库做一些数据库操作，之后再恢复 B 库的网络环境
预期结果	应该可以立刻同步
实际测试结果	但是在 B 库上却看不到同步的数据，通过命令 <code>show slave status\G</code> 查看发现 <code>Slave_IO_Running</code> 的状态是 No，这种状态持续很长一段时间，数据才能同步到 B 库
备注	定位在一个同步延迟相关的参数： <code>--slave-net-timeout=seconds</code> 参数含义：当 B 库从 A 库读取 log 数据失败后，等待多久重新建立连接并获取数据。 于是在配置文件中增加该参数，设置为 60 秒 <code>slave-net-timeout=60</code> 重启 MySQL 数据库后测试，该问题解决

(4) 容错测试用例——Tomcat 自动恢复

容器目前会自动恢复的情况如下：

- Tomcat 的控制台输出出现 OutOfMemory（内存溢出）
- Tomcat 异常关闭



质量全面管控：从项目管理到容灾测试

其他情况目前暂时不提供自动恢复的功能。见表 12-13。

表 12-13 Tomcat 自动恢复

测试用例编号和名称	0004
测试目的	查看 Tomcat 自动恢复功能
前提条件	编写自动恢复脚本，并且已经部署
测试步骤	1. 关闭 Tomcat 进程； 2. 恢复脚本启动——短信告警并自动重启 Tomcat
预期结果	Tomcat 自动恢复
实际测试结果	没有自动恢复
备注	运维人员提出仅仅是内存溢出或者异常关闭，无法全覆盖 Tomcat 的出错情况，还应该包括 UnknowState（状态异常）、logFull（线程池满）、stoped（进程宕掉）、Connection pool exhausted - try increasing（数据库连接池满）这几个关键错误。 同时还定义 Server 端处理机制，要求重启对应的服务，并 email 和 send（如果在一个小时内这台机器重启大于一定数次，则不再重启，直接并 email 和 send 相关人员）

（5）灾难恢复测试用例——验证主机虚拟化。见表 12-14。

表 12-14 验证主机虚拟化

测试用例编号和名称	0005
测试目的	验证虚拟化集群冗余性
前提条件	首先 vSphere 集群已经部署，并支持 vSphere HA 功能，然后集群中有 2 台以上主机，主机上运行有虚拟机和集群主机使用共享存储
测试步骤	1. 通过 VC 登录 vSphere 平台； 2. 从 VC 中手动“重新引导”一台主机（或者拔除所有管理网线及所有光纤线）； 3. 观察被关机主机上的虚拟机是否迁移到集群中的正常主机
预期结果	被关机主机上虚拟机正常迁移到集群中的正常主机，由于被关闭主机是异常关机，所以其上运行的虚拟机将会在集群中的正常主机上重新启动
实际测试结果	虚拟机没有重新启动
备注	PING 虚拟机网卡 IP，将会看到 10 个左右丢包

12.4.3 容灾线上演习

容灾测试进行线上演习，关键点如下：

（1）提前在测试环境做一次充分的容灾测试演练。



(2) 提前准备所有的生产环境的测试数据。因为演练一般从半夜开始，短短的几个小时内需要演练较多的开关。准备充分的测试数据，可以大大提高演练的效率。

(3) 开关切换时间尽量短。虽然是在深夜，也依然有很多用户在使用系统，所以每个开关的切换都要考虑到对当前用户的影响，开关生效时间尽量缩短。

(4) 每个开关切换后都一定要切回并再次验证，必要时演练结束后将所有系统全部重启。

(5) 不同系统之间的同业务的开关执行顺序需要提前沟通。

12.4.4 容灾长期规划

容灾测试可以伴随功能测试一起进行，在每次进行功能验证的过程中，执行一些必要的容灾测试用例。

- 平台功能更新

每当平台需要支撑新的业务时，都会对原有平台功能进行变更，或增加一部分内容，这样的变化称之为“平台功能更新”。当考虑这些变更时，必须同时考虑对其进行容灾测试。

- 新计划制定

在了解平台需要做哪些变更后，一定要对原有的容灾计划进行相应的变更，增加测试用例。

- 测试流程

变更容灾计划后，为了验证其有效性，必须进行测试。若测试失败，将返回到第一步重新开始；若测试成功，那么即可进入正常运行状态。

- 监控平台

根据公司现有的监控平台进行各个指标的监控，通过脚本采集系统的状态，例如 Queue 消息数、Tibco 服务器的状态等。DBA 主要监控数据库，例如 Session 数的报警、bad SQL 的自动 kill 等。

- 定期测试

建议是根据公司具体的实际情况，可以每个月或者每个季度做一次大范围的容灾测试。



参考文献

1. Glenford J. Myers. *Advances in Computer Architecture*, 1st edition, John Wiley & Sons Inc, New York, 1978, Second edition, 1982.
2. Kent Beck. *Implementation Patterns*, 1st edition, Addison-Wesley Professional, Boston, 2007.
3. Alan Page, Ken Johnston, Bj Rollison. *How We Test Software at Microsoft*, 1st edition, Microsoft Press, Redmond, 2008.
4. James A. Whittaker, Jason Arbon, Jeff Carollo. *How Google Tests Software*, 1st edition, Addison-Wesley Professional, Boston, 2012.
5. Cem Kaner. *Lessons Learned in Software Testing: A Context-Driven Approach*, 1st edition, John Wiley & Sons Inc, New York, 2001.
6. Jamie L. Mitchell, Rex Black. *Advanced Software Testing - Vol. 3, 2nd Edition: Guide to the ISTQB Advanced Certification as an Advanced Technical Test Analyst*, 1st edition, Rocky Nook, Santa Barbara, 2011, 2nd edition, 2015.
7. Karl Wiegers, Joy Beatty. *Software Requirements (Developer Best Practices)*, 1st edition, Microsoft Press, Redmond, 1999, 3rd Edition, 2013.
8. J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, Dennis Rea. *Performance Testing Guidance for Web Applications*, Microsoft Press, Redmond, 2007.
9. Charlie Hunt, Binu John. *Java Performance*, Addison-Wesley Professional, Boston, 2011.
10. Bill Venners. *Inside the Java Virtual Machine*, 1st edition, McGraw-Hill Companies, 1997, 2nd edition, 2000.



11. Lisa Crispin, Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*, 1st edition, Addison-Wesley Professional, Boston, 2009.
12. Cédric Beust. *Next Generation Java Testing: TestNG and Advanced Concepts*, Addison-Wesley Professional, Boston, 2012.
13. Elfriede Dustin. *Effective Software Testing: 50 Specific Ways to Improve Your Testing*, Addison-Wesley Professional, Boston, 2002.
14. G. Ann Campbell, Patroklos P. Papapetrou. *SonarQube In Action*, Manning Publications Co., Shelter Island, 2014.
15. Charalampos S. Arapidis. *Sonar Code Quality Testing Essentials*, 1st edition, Packet Publishing Ltd., Birmingham, 2012.
16. Paco Hope & Ben Waltber. *Web Security Testing Cookbook: Systematic Techniques to Find Problems Fast*, 1st Edition, O'Reilly Media, Sebastopol, 2008.

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396; (010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市海淀区万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036

拒绝堆砌臃肿,支持纯正原创

出版事宜请关注  新浪微博 @ 半亩方塘 _

投稿邮箱: sxy@phei.com.cn

加入本书读者QQ群 465398813,以书会友,资源共享!

专家力荐

本书作者凭借多年的测试实战经验,通过此书深入浅出地剖析了测试在整个项目中如何最大化发挥作用。对测试新手而言,本书是一本非常值得阅读的系统性图书。

张翼 掌门一对一创始人

作为一个测试新手,正确、全面地了解测试,并进一步在项目整个周期中合理介入并发挥价值是很重要的,本书就是这样一本源于实战、贯穿体系的优秀读物。

陈霖 厦门云层天咨软件技术有限公司创始人

目前测试工程师需要熟练掌握更多的测试技能,如性能测试、安全测试和自动化测试等,才能成为一名合格的全栈测试工程师。本书作者从实际项目场景出发,结合当下测试工程师所需的测试技能要求,把多年的一线工作经验融入其中,用最清晰的语言带领读者快速步入软件测试技术的前沿。

朴春龙 上海博为峰软件技术股份有限公司副总经理

这是一本测试入门宝典或者新人入职培训手册,介绍了从项目管理、需求评审、单元测试、自动化部署、安全测试、自动化框架设计、性能测试、分析调优、运维监控到容灾测试的一个完整体系,能够帮助读者快速了解项目研发各个阶段的关键步骤,了解每个阶段测试人员都在执行哪些具体工作,以及标准的输入和输出。

本书的实例均来源于项目实践,是在千锤百炼中得到的经验,使用的都是主流的工具。希望读者在阅读完本书之后,可以提升到中高级测试工程师的水平。

本书适合的读者:

- ☒ 从事软件测试工作的初级、中级测试人员;
- ☒ 希望了解项目测试流程的运维人员、项目主管和项目经理;
- ☒ 测试组长、测试经理、质量保证工程师、软件过程改进人员。



博文视点Broadview



新浪微博
weibo.com

@博文视点Broadview



策划编辑: 孙学瑛
责任编辑: 徐津平
封面设计: 李 玲

上架建议: 软件测试

ISBN 978-7-121-30786-7



9 787121 307867 >

定价: 79.00元